ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ
KATEDRA POČÍTAČŮ

Multiplatformní aplikace v React Native pro sledování stavu měřáku kvality ovzduší

Multi-platform application for control of the air quality meter in React Native

BAKALÁŘSKÁ PRÁCE

Studijní program: Otevřená informatika
Studijní obor: Softwarové systémy

Vedoucí práce: Ing. David Sedláček, Ph.D.

Jan Vidašič

Praha 2018

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vidašič** Jméno: **Jan** Osobní číslo: **457120**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Multiplatformní aplikace v React Native pro sledování stavuměřáku kvality ovzduší

Název bakalářské práce anglicky:

Multi-platform application for control of the air quality meter in React Native

Pokyny pro vypracování:

Analyzujte existující mobilní aplikace pro vzdálenou správu elektrického zařízení (např. správa kotle, osvětlení v chytré domácnosti) a definujte seznam požadavků a potřeb uživatelů s ohledem na využití v rámci sledování stavu měřáku kvality ovzduší. Na základě analýzy navrhněte mobilní aplikaci, která umožní sledování stavu a správu měřáku ovzduší.
Navrženou mobilní aplikaci implementujte pomocí multiplatformního frameworku a vyhodnoťte úspěšnost implementace pro obě hlavní platformy, Android a iOS. Při práci postupujte podle metodiky User centered design (UCD) s ohledem na dostupnost cílové skupiny. Pro testování chodu aplikace implementujte generátor realistických dat založený na experimentálním vzorku dodaným vedoucím práce.

Seznam doporučené literatury:

[1] Learning React Native: Building Native Mobile Apps with JavaScript. Bonnie Eisenman. O'Reilly Media, Inc., 2015.
[2] Multiplatformní knihovna pro výukové aplikace. Josef Veselý. Diplomová práce ČVUT, FEL, 2017.
[3] http://www.reactnative.com/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.02.2018** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**

[redacted]
Ing. David Sedláček, Ph.D.
podpis vedoucí(ho) práce

[redacted]
podpis vedoucí(ho) ústavu/katedry

[redacted]
prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

[redacted]
Podpis studenta

Prohlášení:

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

Jan Vidašič

Poděkování

Především bych chtěl poděkovat vedoucímu mé práce Ing. Davidu Sedláčkovi, Ph.D., za jeho rady a ochotu během psaní této práce. Také chci poděkovat všem, kteří se účastnili testování aplikace a poskytli mi tím zpětnou vazbu. Nakonec bych chtěl poděkovat svým rodičům, díky kterým můžu tuto školu vůbec studovat.

Anotace:

Cílem této práce je navrhnout a implementovat uživatelsky příjemnou a intuitivní multiplatformní aplikaci ve frameworku React Native pro systémy iOS a Android, která bude svým uživatelům zobrazovat data ze zařízení měřících kvalitu ovzduší a umožní jim také jejich správu. Z důvodu neexistence daného měřícího zařízení v době psaní práce je zde také navržen a implementován server, databáze a generátor hodnot, který simuluje chování tohoto měřícího zařízení. Aplikace je na závěr otestována s uživateli a na základě jejich odezvy pak přepracována.

Klíčová slova:

Android, aplikace, framework, iOS, multiplatformní, React Native,

Abstract:

The goal of this thesis is to design and implement multiplatform application for iOS and Android systems in the framework called React Native, which will be showing to its users data from the devices measuring air quality and allow them to control these devices. Because of non-existence of this measuring device during writing of this thesis there is also implemented a server, a database and a generator of values whose purpose is to simulate the behavior of said measuring device. At the end of the thesis the application is tested with users and reworked based on their feedback.

Key words:

Android, application, framework, iOS, multiplatform, React Native

Obsah

1. Úvod

Tato práce si klade za cíl navrhnout a implementovat multiplatformní mobilní aplikaci pro vzdálenou správu měřáku stavu a kvality ovzduší. Měřáků budou dva druhy – dražší model s přesným měřením v ceně několik set tisíc, který bude rozmístěn na významnějších lokalitách, a levnější, ne tak přesný, model, který si budou moci běžní lidé pořídit do domácnosti. Aplikace je určena pro druhý model, a kromě správy měřáku bude aplikace jeho data pro uživatele vhodně vizualizovat ve formě aktuálního stavu a grafu s historií naměřených hodnot.

Aplikace typu vzdálené správy se vyznačují přívětivým uživatelským rozhraním, které je intuitivní, uživatele ničím nepřekvapuje a umožňuje mu s minimálním úsilím a v co nejkratším čase dosáhnout jeho záměru. Proto je třeba klást v návrhu i implementaci na uživatelské rozhraní velký důraz a dodržovat standardy a doporučení pro danou platformu (iOS či Android).

Úvodní, analytická, část práce se zabývá kvalitou ovzduší a jejím vlivem na zdraví, a dále rešerší již existujících aplikací pro měření stavu ovzduší. Další kapitola se pak věnuje návrhu nejen vlastní aplikace s ohledem na cílovou skupinu uživatelů a jejich požadavky, ale také serveru, databáze a generátoru hodnot. Daný měřák ovzduší totiž zatím existuje pouze jako prototyp, bylo tedy potřeba jeho chování simulovat. Následující část práce pojednává o samotné implementaci jak této aplikace, tak generátoru realistických dat a serveru s databází, který tato data bude aplikaci zprostředkovávat. Na implementační část práce pak navazuje kapitola o testování, kde je aplikace otestována s uživateli na iOS i Android zařízení.

2. Analýza

Kapitola o analýze v první části rozebírá kvalitu ovzduší a její důsledky na lidské zdraví. V druhé části je pak provedena rešerše již existujících aplikací pro měření stavu ovzduší.

2.1 Kvalita ovzduší a její měření

V této kapitole je popsáno, jaké druhy znečištění ovzduší existují, jeho vliv na lidské zdraví, a nakonec co vlastně bude měřit zařízení, které bude spojeno s naší mobilní aplikací.

2.1.1 Typy znečištění ovzduší

Znečištění ovzduší je rozděleno, dle toho kde se nachází, na znečištění v domácnosti a znečištění venkovní a dále, dle typu, na plynné a částicové. Nejčastější znečištění v domácnosti je způsobeno kouřem, který vzniká při zatápění či vaření, částicemi uvolňujícími se z domovní izolace a stavebních materiálů, z nichž nejnebezpečnější je rakovinotvorný azbest, a dále plísňovými sporami a přírodně se vyskytujícím radonem. Největším viníkem venkovního znečištění zaviněným lidmi jsou motorová vozidla, továrny, spalování paliv za účelem výroby elektřiny nebo zatopení v domě. Níže jsou popsány látky s nejškodlivějším vlivem na lidské zdraví dle WHO [1].

- PM10 – Částice menší než 10 μm. Jedná se o soli, černý uhlík, amoniak, minerální prach, dusičnany a sírany. Jejich nebezpečí tkví v schopnosti proniknout do dýchacích cest a usazovat se v plicích.
- PM2.5 – Částice menší než 2.5 μm. Typ částic (černý uhlík, minerální prach atp.) je stejný jako u částic PM10. Jejich hlavním zdrojem jsou spalovací motory a spalování uhlí a biopaliv. Po vdechnutí se usazují nejen v plicích, ale jsou schopné proniknout i do krevního oběhu.
- PM1 – Částice menší než 1 μm. Jedná se o totožný typ částic se stejným vlivem na zdraví jako částice PM2.5.
- O_3 – Jedná se o přízemní ozón. Vzniká chemickou reakcí v ovzduší. Hlavní roli v této reakci hrají emise z výfukových plynů, metan, oxid uhelnatý a sluneční světlo, tedy zvýšené koncentrace ozonu vznikají za slunečného počasí. Způsobuje astma a zánětlivá onemocnění plic.
- NO_2 – Oxid dusičitý. Jeho zdrojem jsou spalovací motory a elektrárny. Zhoršuje symptomy bronchitidy, astma a způsobuje infekce dýchacích cest.
- SO_2 – Oxid siřičitý. Vzniká při spalování fosilních paliv a při tavení rud. Zdravotní rizika má podobné jako oxid dusičitý. Při kombinaci s vodou ve vzduchu způsobuje kyselý déšť.

2.1.2 Vliv na zdraví

V roce 2012 mělo znečištění ovzduší celosvětově za následek asi 11.6 % všech úmrtí [2], patří tak mezi významná zdravotních rizika, a proto se WHO snaží o prosazování jejich omezování v jednotlivých státech světa. Mezi nemoci, které má znečištění ovzduší na svědomí, patří mrtvice, rakovina plic, infekce dolních dýchacích cest, CHOPN a ischemická choroba srdeční.

V roce 2012 bylo asi 35 % všech případů mrtvic a asi 9 % případů rakoviny plic zaviněno špatnou kvalitou vzduchu [2]. Velké procento těchto smrtí se týká rozvojových zemí a je zapříčiněno znečištěním ovzduší v domácnosti, kde lidé denně používají při vaření otevřený oheň a jednoduchá kamna a sporáky se dřevem a uhlím jako palivem.

2.1.3 Částice monitorované měřákem a jejich limity

Měřák spojený s naší aplikací bude měřit znečištění ovzduší prachovými částicemi PM1, PM2.5 a PM10. Téměř každý z těchto typů částic má své limity, o kterých je třeba informovat i uživatele aplikace. Tyto limity mají různé státy nastaveny jinak, nicméně v aplikaci jsme se rozhodli řídit doporučeními WHO [3]. Přehled limitů je v Tabulka 1.

Tabulka 1: Prachové částice a jejich limity

Název typu částice	Limit – roční průměr [$\mu g/m^3$]	Limit – průměr za 24 hodin [$\mu g/m^3$]
PM10	20	50
PM2.5	10	25
PM1	nestanoven	nestanoven

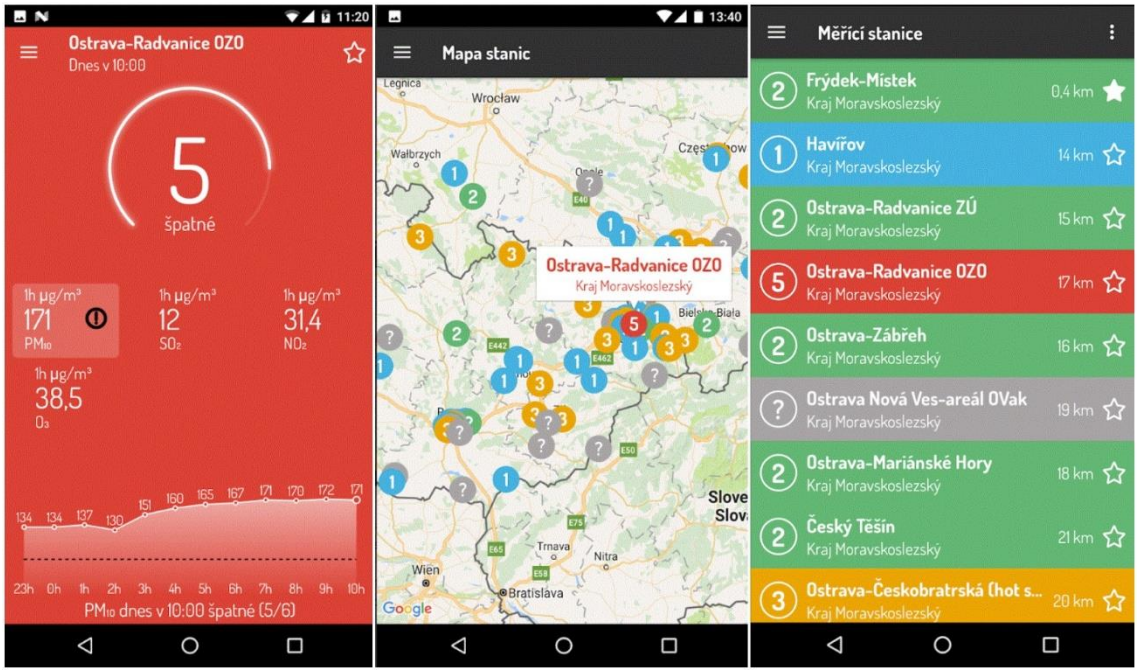
Pro částice PM1 doporučené limity neexistují, jejich výskyt ale koreluje s výskytem částic PM2.5 a PM10 a účinek na zdraví je podobný jako u částic PM2.5. Pro účely aplikace jsme proto zvolili pro PM1 stejné limity jako u PM2.5.

2.2 Rešerše existujících řešení

Pro rešerši byly vybrány dvě aplikace, které se té naší tematicky a funkčně velmi blíží, tedy také se týkají kvality ovzduší a jejího měření. První aplikace pouze zobrazuje naměřená data z webu, ta druhá funguje s vlastním měřákem, se kterým komunikuje.

2.2.1 SmogAlarm

Aplikace SmogAlarm[1] od společnosti Čisté nebe, o.p.s. se nachází na platformách iOS i Android a zobrazuje údaje o kvalitě vzduchu (koncentrace prachu, oxidu siřičitého, uhelnatého, celkový index kvality), které získává ze stanic evidovaných ČHMÚ. Uživatel si může zobrazit přehled těchto stanic, filtrovat je například podle vzdálenosti od jeho lokace, podle kvality ovzduší nebo podle názvu. Každá stanice v tomto přehledu má barevné pozadí podle indexu kvality stavu ovzduší, kde barvy odpovídají barvám na webových stránkách[2] ČHMÚ. Každou stanici je možno přidat do oblíbených, nebo rozkliknout a zobrazit všechny dostupné údaje včetně indexu kvality a jeho statistiky za posledních 12 hodin. V aplikaci lze nalézt informace o jednotlivých částicích, jejich zdravotní rizika a limity. V neposlední řadě si uživatel může také zobrazit mapu všech stanic, včetně indikátoru jejich indexu kvality stavu ovzduší. Na Obrázek 1: Obrazovky aplikace SmogAlarm jsou zobrazena hlavní okna aplikace SmogAlarm.



Obrázek 1: Obrazovky aplikace SmogAlarm

Aplikaci lze co se týče designu těžko něco vytýkat. Svůj účel splňuje s lehkostí, uživatel má všechny důležité informace přehledně dostupné. Jedinou výtkou může být tzv. „hamburger menu“ v levé horní části aplikace, které slouží k navigaci. Toto menu, na platformě Android časté, se vyskytuje i na iOS verzi. Dle metodik Apple, týkajících se uživatelských rozhraní iOS aplikací (je ale vhodné je dodržovat na všech platformách), není takové řešení doporučené. Důvody, proč tomu tak je, jsou v následujícím seznamu. [4]

- Uživatel na první pohled často neví, v jaké části aplikace se nachází (není případ této aplikace, název aktuální části aplikace se v horní části obrazovky zobrazuje).

[1] https://play.google.com/store/apps/details?id=cz.ubik.smogalarm&hl=en

[2] http://portal.chmi.cz/files/portal/docs/uoco/web_generator/actual_hour_data_CZ.html#legend

- Uživatele toto menu zdržuje. Pro přechod do jiné části aplikace musí zvolit ikonku menu, najít hledanou část a až poté na ni přejít, což je také jeden z důvodů, proč se pak použití tohoto typu menu nejraději vyhnou.
- Ikonka menu je na místě tlačítka zpět, které se v aplikacích často vyskytuje a je potřeba, pokud se dostaneme na obrazovku, která není součástí menu (to je problém i na Android verzi této aplikace, kde uživatel rozklikne detail stanice a aby se dostal zpět, musí znova projít procesem použití ikonky menu a volby předchozí části aplikace – přehledu stanic).
- Nakonec, protože je v takovém typu menu spousta místa, může být problémem to, že svádí vývojáře přidávat do něj spoustu nadbytečných položek.

Protože je naše aplikace multiplatformní a měla by dodržovat standardy obou platforem, bude v ní využito řešení menu typu Tab Bar[3], ve kterém jsou jednotlivé části menu ve spodní části obrazovky jako panel.

2.2.2 Kanárci

Aplikace Kanárci je součástí projektu Kanárci.cz[4] a vznikla roku 2014 na platformě iOS jako diplomová práce[5] Jana Remeše na Fakultě elektrotechnické ČVUT. Kanárek je zařízení na měření kvality ovzduší, který se s telefonem spáruje pomocí audio portu nebo Bluetooth a měření pak probíhá automaticky – aplikace Kanárka neovládá. S Kanárkem může uživatel chodit a trasa společně s naměřenými údaji se v aplikaci zobrazuje a ukládá na server. Menu se skládá z pěti hlavních záložek popsaných v seznamu níže.

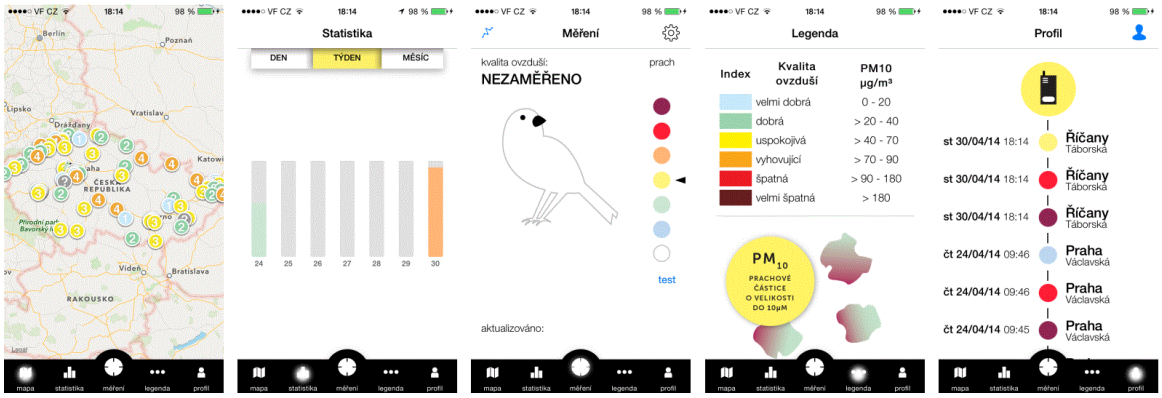
- Záložka s mapou kvality ovzduší České Republiky, která je prakticky totožná s mapou v aplikaci SmogAlarm a data také čerpá z ČHMÚ, navíc ale ukazuje data z dostupných kanárků.
- Záložka statistik měření za poslední den, týden a měsíc. Statistiky jsou realizovány pomocí sloupcového grafu a každý sloupec je zabarven dle nejhorších dosažených hodnot v daný den.
- Záložka měření, ve které se zobrazuje kanárek zabarvený podle aktuálního stavu ovzduší a je zde také nastavení způsobu připojení kanárka či spuštění záznamu trasy.
- Záložka s legendou, kde uživatel najde informace o prachových částicích a jejich limity. Legenda je formou webové stránky, která je uložena na serveru kanarci.cz a v aplikaci je její, při každém načtení legendy aktualizující se, kopie.
- Záložka s historií měření, kde má uživatel přehled všech svých provedených měření a může si rozkliknout jejich detail.

Vzhled aplikace Kanárci je zobrazen na Obrázek 2: Obrazovky aplikace Kanárci.

[3] https://developer.apple.com/ios/human-interface-guidelines/bars/tab-bars/

[4] http://kanarci.cz/

[5] https://dspace.cvut.cz/handle/10467/24951



Obrázek 2: Obrazovky aplikace Kanárci

Aplikace Kanárci se už velmi blíží tomu, jak by mohla vypadat ta naše. Hamburger menu z aplikace SmogAlarm zde vystřídal ovládací prvek Tab Bar a v testovací části této diplomové práce je uvedeno, že uživatelé tento typ navigace ocenili [5].

3. Návrh aplikace

V úvodní části kapitoly je rozebrána cílová skupina, které se bude aplikace týkat, dále pak požadavky na aplikaci s ohledem na tuto cílovou skupinu a požadavky zadané vedoucím práce. Rovněž jsou zde popsány případy užití, architektura aplikace, a nakonec návrhy obrazovek.

3.1 Cílová skupina

Cílovou skupinou jsou uživatelé, které zajímá stav ovzduší v jejich sousedství nebo domě. ČHMÚ totiž poskytuje pouze data z okresů, ale lokálně se tyto hodnoty mohou velmi lišit (sousedé zatápí v kamnech nevhodným způsobem, dům se nachází v blízkosti silnice či továrny, v domě jsou kamna či kotel atp.). Jedná se tedy o cílovou skupinu dospělých lidí všech věkových kategorií a v návrhu aplikace se proto musí počítat i se staršími lidmi, kteří nemusí být velmi zruční v ovládání mobilního telefonu a uživatelské rozhraní by tak mělo být co nejjednodušší. Počítá se s tím, že k měřáku vznikne webová stránka, která bude obsahovat podrobnější informace a statistiky a zkušenějším uživatelům se zájmem o větší detail (týká se především grafů) se tedy aplikace nemusí přizpůsobovat a může zachovat jednoduchost vzhledu.

3.2 Požadavky

Základní požadavky byly zadány vedoucím práce a na základě analýzy cílové skupiny a vlastního rozmyšlení pak byly autorem práce přidány další. Jelikož je součástí této práce vytvoření nejen aplikace, ale i databáze, serveru a generátoru dat (který simuluje chod měřáků), jsou zde požadavky rozděleny podle toho, jakého systému se týkají. Požadavky jsou rozděleny do dvou kategorií – funkční a nefunkční (kvalitativní). Funkční požadavky jsou obecně požadavky, které popisují, co bude systém dělat. Kvalitativní pak popisují, jak to bude dělat.

3.2.1 Funkční požadavky na aplikaci

Systém (aplikace):

- bude zobrazovat aktuální data z měřáku.
- umožní uživateli přidat měřák a přidaný měřák se uloží do paměti telefonu.
- bude zobrazovat graf historických hodnot z měřáku dle uživatelem zadaného časového intervalu (den, týden, měsíc).
- bude zobrazovat informace o částicích, které jsou měřeny, a jejich limitech.
- umožní uživateli zapnout a vypnout měřák.
- umožní uživateli smazat měřák z aplikace a paměti telefonu.

3.2.2 Funkční požadavky na server

Systém (server):

- umožní aplikaci získat data daného měřáku.
- bude přijímat a ukládat data generátoru.
- umožní aplikaci ověřit existenci měřáku s daným názvem pro účely spárování s jeho daty na serveru.

3.2.3 Funkční požadavky na generátor hodnot

Systém (generátor):

- bude generovat realistické hodnoty prachových částic v ovzduší společně s časem vygenerování a jménem měřáku, který je naměřil, a následně je zašle na server.

3.2.4 Kvalitativní požadavky na aplikaci

- Aplikace bude fungovat na operačních systémech iOS a Android.
- Uživatel bude moci přidat měřák načtením jeho QR kódu či ručním zadáním jeho názvu.

3.2.5 Kvalitativní požadavky na server

- Server bude s klienty (aplikací a generátorem) komunikovat pomocí REST rozhraní.
- Server bude data generátoru ukládat a číst z databáze.
- Přenášená data mezi aplikací a serverem budou ve formátu JSON (z důvodu jednodušší práce s daty a větší rychlost při jejich přenosu než XML).

3.2.6 Kvalitativní požadavky na generátor hodnot

- Generátor bude vygenerovaná data posílat na server každých 15 sekund.

3.3 Případy užití

Případy užití jsou vizuálním zobrazením funkčních požadavků na systém. Popisují interakci mezi aktérem a systémem, kde aktér má nějaký cíl, kterého chce dosáhnout. Aktérem může být osoba, ale například také jiný systém, případně čas. Případy užití jsme omezili na server a aplikaci společně s jejich aktéry a funkčními požadavky.

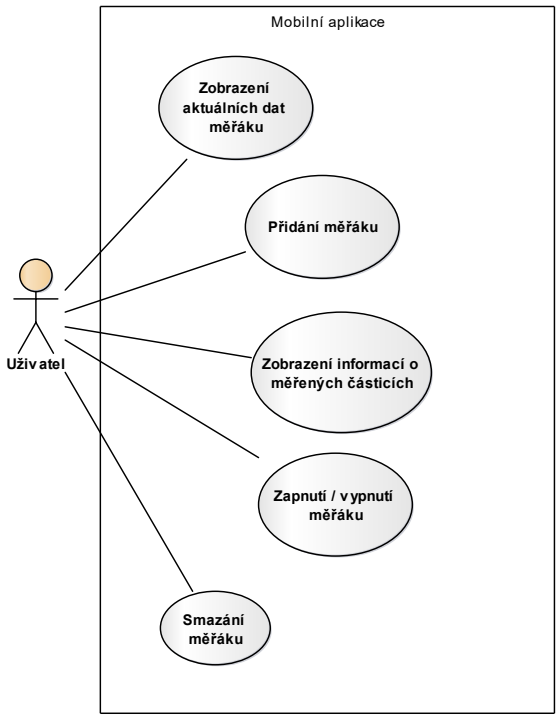
3.3.1 Aktéři případů užití

Aktéři vyskytující se v našich případech užití jsou vypsáni v seznamu níže.

- Uživatel – Jedná se o uživatele aplikace.
- Aplikace – Naše mobilní aplikace, která komunikuje se serverem.
- Generátor – Generátor dat, který komunikuje se serverem.

3.3.2 Případy užití aplikace

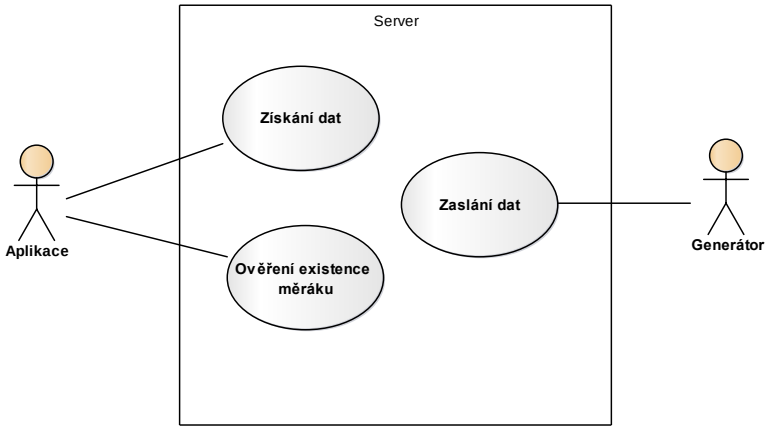
Případy užití aplikace jsou znázorněny na Obrázek 3: Případy užití aplikace.



Obrázek 3: Případy užití aplikace

3.3.3 Případy užití serveru

Případy užití serveru jsou znázorněny na Obrázek 4: Případy užití serveru.



Obrázek 4: Případy užití serveru

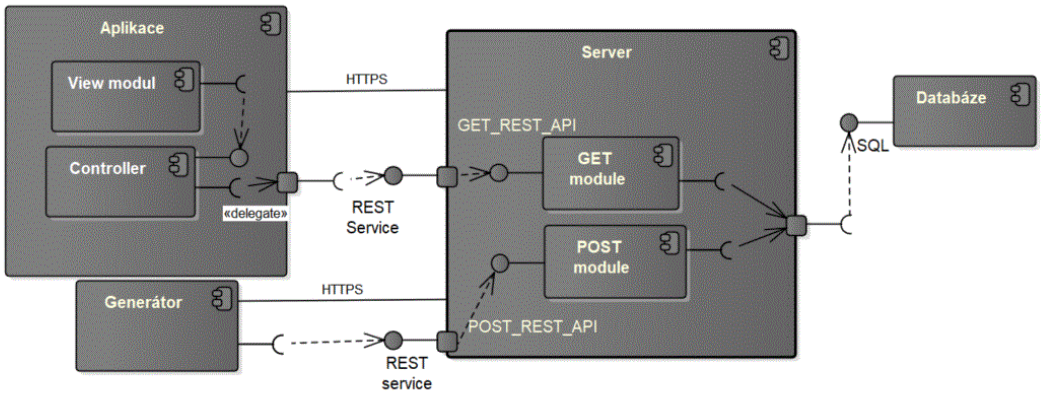
3.4 Návrh architektury

V této kapitole je navržena a popsána celková architektura systému s jeho komponentami, jimiž jsou samotná aplikace, generátor dat a server s databází.

3.4.1 Diagram komponent

Diagram komponent popisuje, jak spolu jednotlivé komponenty systému komunikují. Aplikace má v sobě dvě hlavní vrstvy. Vrstvu View, která zobrazuje uživatelské rozhraní a vrstvu Controller, která zprostředkovává komunikaci se serverem pro vrstvu View. Server poskytuje klientům REST rozhraní, které se dělí na dva moduly. První modul obsluhuje HTTP dotazy typu GET (HTTP požadavek na data z daného zdroje), které server dostává od Controlleru mobilní aplikace vždy, když aplikace aktualizuje data měřáku, nebo uživatel přidává zařízení. Tento

modul slouží ke čtení dat z databáze. Druhý modul obsluhuje HTTP dotazy POST (HTTP požadavek sloužící pro posílání dat), které server dostává od generátoru dat. Tento modul slouží k zápisu dat do databáze. Diagram komponent se nachází na Obrázek 5: Diagram komponent.



Obrázek 5: Diagram komponent

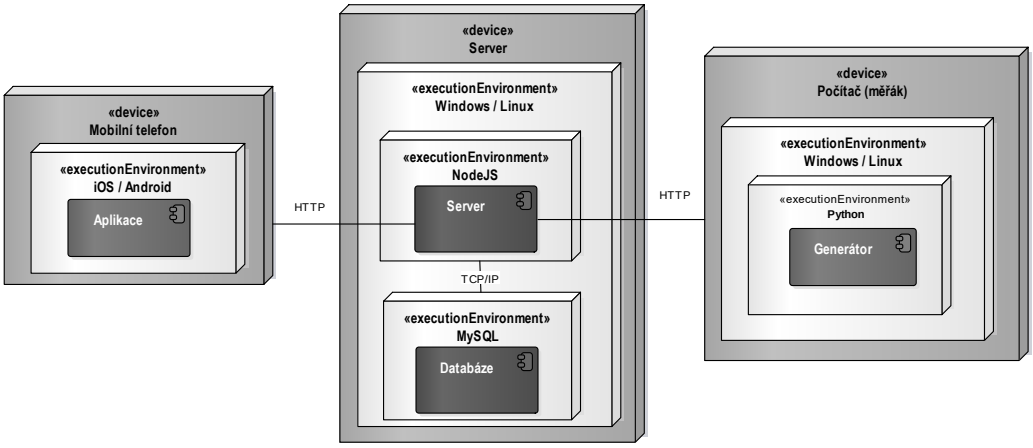
3.4.2 Diagram nasazení

Diagram nasazení popisuje jednak hardwarovou architekturu systému, ale také operační systémy a softwarové prostředí běžící na jednotlivých zařízeních.
Náš systém se skládá ze zařízení popsaných v seznamu níže.

- Mobilní telefon s operačním systémem iOS či Android, na kterém běží naše aplikace, která je psaná dle zadání vedoucího práce v jazyce React Native[6].
- Počítač, který slouží jako server s databází MySQL[7] (ta by ale mohla být i na vlastním zařízení) a kde byl pro řešení serveru zvolen Node.js[8] na bázi jazyka JavaScript. Protože server i databáze jsou jednoduché a nebudou na ně kladeny velké zátěžové nároky, výběr jazyka a prostředí nehraje roli.
- Počítač, který slouží jako generátor dat a simuluje funkci měřáku. Pro vytvoření generátoru byl zvolen skriptovací jazyk Python.

Diagram nasazení se nachází na Obrázek 6: Diagram nasazení.

[6] https://facebook.github.io/react-native/
[7] https://www.mysql.com/
[8] https://nodejs.org/en/



Obrázek 6: Diagram nasazení

3.5 Návrh obrazovek aplikace

Návrhy vzhledu obrazovek (někdy též zvané wireframes) byly vytvořeny pomocí modelovacího nástroje proto.io[9]. Vytvořeny byly pouze stěžejní obrazovky aplikace jako vzor pro implementaci, důraz byl kladen na jednoduchost vzhledu a ovládacích prvků.

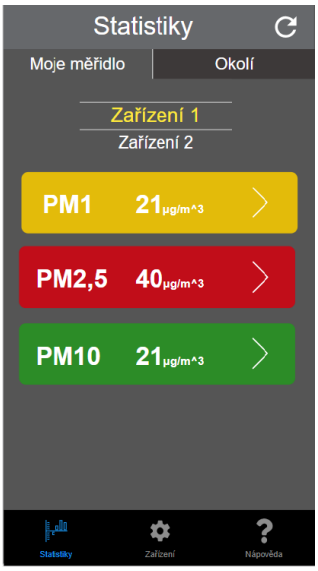
3.5.1 Obrazovka statistik

Jedná se o hlavní obrazovku aplikace, protože zobrazuje poslední naměřená data měřáku. Její návrh se nachází na Obrázek 7: Návrh obrazovky statistik. Ve spodní části se nachází navigace Tab Bar, v horní části pak název aktuální obrazovky, ikona aktualizace údajů a další Tab Bar, který slouží k přepínání mezi údaji z uživatelova měřáku a jeho okolím (například okres či kraj). Prostřední část se skládá z Pickeru[10], který slouží k výběru uživatelova měřáku, a ListView[11], který zobrazuje naměřené hodnoty daného měřáku a každá jeho položka je podbarvena dle aktuálního stavu. Barvy budou tři a budou se řídit tím, zda daná hodnota koncentrace daného typu částic je pod limitem, překročila doporučený roční průměr nebo překročila limit průměru za 24 hodin. Tyto limity byly popsány v kapitole 2.1.3. Každá položka jde také rozkliknout, zobrazí se tím obrazovka s detailními údaji včetně grafu – tato obrazovka se nachází v následující kapitole 3.5.2.

[9] https://proto.io/

[10] https://developer.android.com/guide/topics/ui/controls/pickers

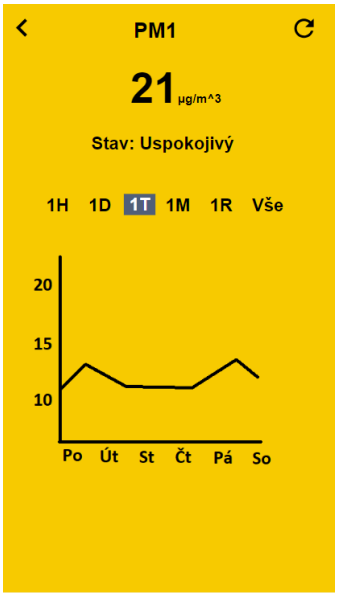
[11] https://developer.android.com/guide/topics/ui/layout/listview



Obrázek 7: Návrh obrazovky statistik

3.5.2 Obrazovka detailních statistik

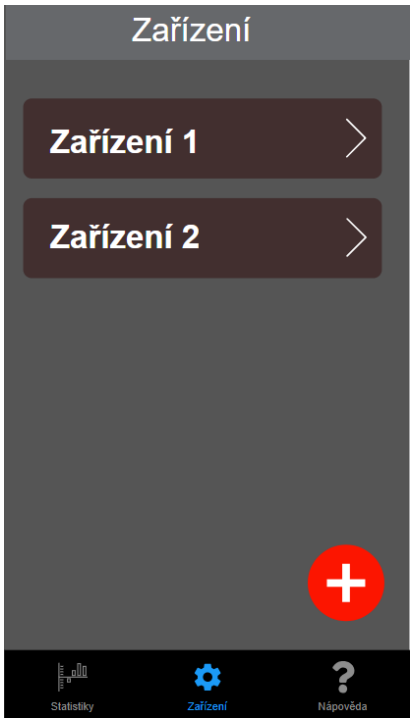
Obrazovka obsahuje detailní údaje o daném typu částice, aktuální stav a graf s možností výběru časového období. Návrh její podoby je na Obrázek 8: Návrh obrazovky grafu. Pozadí obrazovky je stejné jako podbarvení odpovídající položky ListView na obrazovce statistik v kapitole 3.5.1.



Obrázek 8: Návrh obrazovky grafu

3.5.3 Obrazovka měřáků

Obrazovka obsahuje ListView, kde položky jsou jednotlivé měřáky. Každý měřák jde rozkliknout a zobrazí se obrazovka s jeho ovládáním. Ve spodní části obrazovky se nachází tlačítko pro přidání měřáku, po kliknutí na toto tlačítko by měl uživatel mít možnost zvolit mezi načtením QR kódu a ručním zadáním zařízení. Návrh této obrazovky se nachází na Obrázek 9: Návrh obrazovky přehledu zařízení uživatele.



Obrázek 9: Návrh obrazovky přehledu zařízení uživatele

4. Implementace

V této kapitole je popsána implementace mobilní aplikace, serveru, databáze a generátoru dat, společně s knihovnami a technologiemi použitými k jejich tvorbě.

4.1 Mobilní aplikace

V úvodních částích kapitoly jsou nejprve rozebrány použité technologie a knihovny a následně pak jednotlivé obrazovky aplikace včetně jejich fungování a vzhledu.

4.1.1 Nativní multiplatformní aplikace

Každý mobilní operační systém má vlastní SDK, které je psané v určitém jazyce (Java pro Android, Objective-C/Swift pro iOS). Nicméně obvykle je nad tímto SDK také postaveno API, které pak mohou využívat i jiné jazyky. Na Android i iOS toto API existuje, a proto je možné psát jednu aplikaci s jedním zdrojovým kódem, který pak daný framework určený pro multiplatformní vývoj převede díky SDK API na nativní aplikaci dané platformy. Mezi takové frameworky patří například Xamarin nebo React Native, který byl také použit pro vývoj naší aplikace a je popsán v další kapitole.

4.1.2 React Native

Vedoucím práce byl pro vytváření aplikace zvolen React Native. Je to open-source framework vytvořený společností Facebook a zveřejněný v roce 2015. Umožňuje vyvíjet nativní multiplatformní aplikace pomocí jazyka JavaScript a frameworku React, který se mimo jiné používá pro vývoj webových aplikací. Tato práce používá React Native verze 0.55, což je v době jejího psaní také nejnovější verze. React Native podporuje operační systémy Android verze 4.1 (API 16) a vyšší a iOS verze 8.0 a vyšší.

Základním prvkem Reactu a React Native jsou komponenty (components). Každá tato komponenta má svoje *props* (vlastnosti) a *state* (stav). Props dané komponenty se nastavují při jejím vytváření, a zůstávají stejné po celou dobu jejího života. Pro data komponenty, která se budou v komponentě měnit je zde *state*. *State* se inicializuje v konstruktoru komponenty a v momentě kdy jej chceme změnit, zavoláme metodu *setState.* Tato metoda je asynchronní a kvůli výkonu se může React rozhodnout její vykonání odložit. Po změně stavu je vždy automaticky zavolána funkce *render*. Funkce *render* slouží k vykreslení komponenty. Uvnitř *render* jsou definovány UI komponenty pomocí značkovacího jazyka JSX. Podoba syntaxe JSX a funkce *render* je zobrazena na Obrázek 10: Ukázka funkce render a syntaxe JSX. [6]

```
21
22  render() {
23    const { navigate } = this.props.navigation;
24    return (
25      <View style={styles.container}>
26          <View style={styles.inputWrap}>
27            <TextInput
28              style={styles.input}
29              returnKeyType={'done'}
30              onChangeText= {(text) => deviceName = text}
31              value={null}
32            />
33          </View>
34          <View style={styles.buttonWrap}>
35              <TouchableHighlight style={styles.button} onPress={this.setDevice.bind(this)}>
36                <Text style={styles.nadpis}>Přidat toto zařízení</Text>
37            </TouchableHighlight>
38          </View>
39      </View>
40    );
41  }
```

Obrázek 10: Ukázka funkce render a syntaxe JSX

Základem JSX jsou komponenty jako View, Text nebo Image. Tyto komponenty jsou následně frameworkem mapovány na nativní komponenty dané platformy. Vzhled a rozložení jednotlivých komponent se nastavuje pomocí stylů, které jsou založeny na CSS.

4.1.3 Expo

Při vytváření React Native aplikací má vývojář možnost se rozhodnout ze dvou možností dle toho, zda bude chtít v aplikaci psát nativní kód nebo se bez něj obejde. První možnost vyžaduje instalaci Android Studia, Xcode a dalších nástrojů. Druhou možností je použití projektu Expo, který se bez Android Studia a Xcode obejde a spoustu věcí řeší za nás. Pro vývoj naší aplikace byl použit projekt Expo[12], což je nástavba React Native. Skládá se ze dvou částí. Tou první je Expo XDE, které slouží k vytváření nových React Native projektů a umožňuje vytvořit na počítači instanci projektu, která je dostupná pro zařízení na stejné síti. Druhá část Expo je mobilní aplikace Expo Client na App Store či Google Play, která umožňuje tuto instanci projektu spustit na mobilním telefonu. Při jakékoliv změně v kódu lze pak vidět okamžitě změny i v aplikaci na mobilním telefonu. Expo již v dnešní době podporuje přístup k většině důležitých modulů a API zařízení díky Expo SDK. Pro naši aplikaci nebyl potřeba žádný nepodporovaný modul či nutnost psát vlastní nativní kód, nicméně pokud tato potřeba v projektu vznikne, má vývojář možnost projekt od Expo odpojit.

4.1.4 Stuktura projektu

Seznam níže popisuje složky projektu.

- kořenový adresář
  - .expo
  - .git
  - assets
  - node_modules
  - src

V kořenovém adresáři se kromě vypsaných složek nachází také soubor *App.js*, který je kořenovým souborem aplikace. Další důležitý soubor je *package.json*, který obsahuje závislosti (např. knihovny) a jejich verze.

Zdrojový kód práce se nachází ve složce *src*. Struktura této složky je v seznamu níže.

- src
  - Components
  - Constants
  - Controllers
  - Redux

Složka Components je vrstvou View (uživatelské rozhraní) a obsahuje komponenty představující obrazovky aplikace, které budou rozebrány v kapitole 4.1.6 a dva další soubory navíc. Jedním z nich je *App.js*, který je vstupním souborem aplikace při jejím spuštění. Nastavuje stav aplikace, zadává pokyny ke stahování dat ze serveru. Dalším souborem je *navigator.js,* který je renderován souborem *App.js.* Vytváří navigaci aplikace a mapuje všechny obrazovky.

Složka *Constants* obahuje jediný soubor – *constants.js.* V tomto souboru jsou definovány IP adresy serveru, a také například limity jednotlivých částic znečištění ovzduší.

[12] https://expo.io/

Složka Controllers obsahuje soubory, které představují vrstvu Controller. Tato vrstva obsahuje funkce, které potřebuje používat více komponent. Ve složce se nachází 3 soubory. Soubor *serverController.js* obsluhuje komunikaci komponent se serverem, *appController.js* má na starost vyhodnocování hodnot částic znečištění ovzduší a *storageController.js* se stará o ukládání a načítání hodnot z paměti telefonu.

Složka Redux obsahuje dva soubory – *actions.js* a *reducers.js.* Jejich smysl je vysvětlen v kapitole 4.1.5.2.

4.1.5 Hlavní použité knihovny a API

V této kapitole jsou popsány nejdůležitější knihovny React Native, které byly použity při implementaci aplikace. Do aplikace se dají importovat pomocí *npm*[13], což je mimo jiné registr JavaScript knihoven. Ukázka importu: *import {StackNavigator} from 'react-navigation'*.

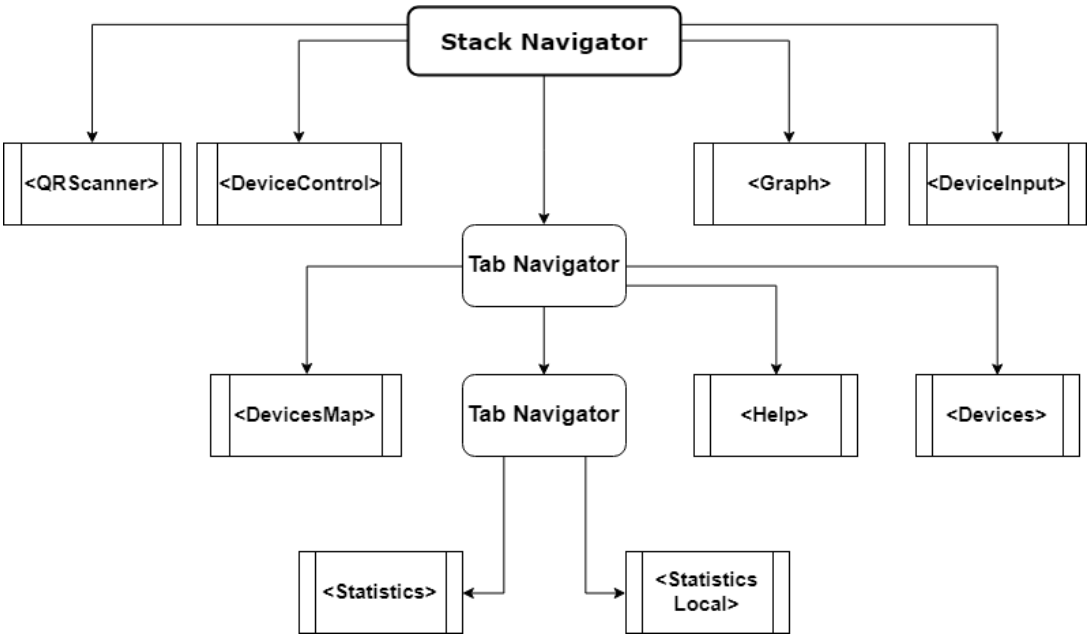
4.1.5.1 React Navigation

Pro navigaci v rámci aplikace byla využita knihovna React Navigation[14]. V naší aplikaci se vyskytují 2 typy navigace. Tou první je stack navigace (*Stack Navigator*), která funguje na bázi zásobníku, tedy disponuje operacemi push, která vloží na zásobník obrazovku, na kterou jsme právě přešli a operací pop, která vrchní obrazovku ze zásobníku odstraní a vrátí se zpět na tu předchozí. Aktuální obrazovka je vždy na vrcholu zásobníku. Druhým typem navigace je tab navigace (*Tab Navigator*), která se skládá z horní nebo dolní lišty záložek představující obrazovky, mezi kterými může uživatel přecházet doleva či doprava. Naše aplikace má hlavní navigaci typu stack, dále vnořenou navigaci typu tab a v ní vnořenou navigaci také typu tab. Na Obrázek 11: Struktura aplikace a navigace je přehled všech komponent společně s navigacemi pro lepší pochopení.

Z hlediska uživatele je hlavní navigací první vnořený *Tab Navigator*, který má formu spodní lišty záložek a obsahuje stěžejní obrazovky aplikace – přehled zařízení (*Devices*), nápovědu (*Help*), mapu všech dostupných zařízení (DevicesMap) a obrazovku statistik, kterou představuje druhý vnořený *Tab Navigator*. Ten obsahuje horní lištu záložek, ve které lze přepínat mezi statistikami uživatelova zařízení (*Statistics*) a lokálními statistikami (*StatisticsLocal*). Z některých obrazovek, které jsou součástí tab navigace, pak vedou odkazy na obrazovky stack navigace. Těmito obrazovkami, na které vedou odkazy, jsou čtečka QR kódu (*QRScanner*), ovládání zařízení (*DeviceControl*), graf (*Graph*) a obrazovka pro ruční vložení kódu zařízení (*DeviceInput*). Z obrazovek se uživatel může vrátit, stiskem šipky zpět v horní části obrazovky nebo stisknutím tlačítka zpět telefonu, na hlavní tab navigaci.

[13] https://www.npmjs.com/

[14] https://github.com/react-navigation/react-navigation



Obrázek 11: Struktura aplikace a navigace

4.1.5.2 Redux a globální správa stavu aplikace

V React Native má každá komponenta svůj stav (*state*). Někdy je ale potřeba, aby na změnu tohoto stavu reagovaly i jiné komponenty. To se dá různě obcházet, například předáváním callback funkcí mezi komponentami, nicméně čím větší je aplikace, tím se přehlednost jejího fungování zmenšuje. Řešením tohoto problému se zabývá například Redux[15], který byl použit v této práci. Jedná se o knihovnu, která slouží k udržování globálního stavu aplikace v takzvaném *store*, do kterého má přístup každá komponenta, která si *store* importuje. Vzhledem k tomu, že se v aplikaci vyskytují 2 typy stavů – globální a stav komponenty, je třeba si rozmyslet, který stav nechat pouze v komponentě a který ukládat do *store*. Redux má tři hlavní principy [7].

- Jediný zdroj pravdy – stav celé aplikace je uložen v jediném objektovém stromu, který se nachází ve *store*.
- Stav aplikace je jen pro čtení – nejde tedy přímo měnit. Pro změnu stavu je potřeba vyslat *action* (akci). Tyto změny jsou centralizované a vykonávají se jedná po druhé, nehrozí tedy riziko konfliktu.
- Změny se vykonávají pomocí *pure* funkcí (jedná se o funkce, které pro stejný argument vrátí vždy stejný výsledek, přičemž nevykonávají žádná vedlejší volání a závisí jen na argumentech). Těmto funkcím se v kontextu knihovny říká *reducers.* Dle typu *action* se pak vybere příslušný *reducer,* který vezme původní stav aplikace a vrátí objekt s novým stavem, aniž by původní objekt jakkoliv měnil.

Komponenty mohou data ze *store* nejen číst, ale mají možnost využít funkci *store.subscribe(callback)*. Funkcí *subscribe* se přihlásí k jakýmkoliv změnám, které ve *store* proběhnou a při jakékoliv změně je automaticky zavolána funkce *callback.* Komponenty tak mají možnost na změny ve *store* reagovat například změnou svého stavu a následného implicitního vyvolání funkce *render,* který komponentu znova překreslí. Komunikace komponent a serveru pak v naší aplikaci probíhá pevně daným způsobem v pořadí, které je znázorněno v následujícím seznamu.

[15] https://redux.js.org/

1. Komponenta zavolá funkci server controlleru a ten přepošle dotaz na server.
2. Server vrátí server controlleru vyžádaná data a controller vyšle danou *action* (akci) pro uložení těchto dat do *store* a ty jsou následně uložena.
3. *Store* vyšle signál všem posluchačům (komponentám), že se jeho stav změnil.
4. Komponenta reaguje na změnu ve *store*.

V naší aplikaci se do *store* ukládají všechna zařízení, které uživatel přidá, společně s jejich statistikami a historickými údaji. Dále lokální zařízení s jejich statistikami (například okres / kraj) a historickými údaji a všechna dostupná zařízení (globálně, nejen uživatelova zařízení) společně s jejich aktuálními údaji a GPS pozicemi. Nakonec se do *store* také ukládá, které zařízení bylo zvolené uživatelem na obrazovce aktuálních statistik jako poslední.

4.1.5.3 Victory Native

Victory Native[16] je knihovna pro vykreslování různorodého množství grafů a kontejnerů, které se spolu dají různě kombinovat. V aplikaci byly využity komponenty typu *VictoryChart* (kontejner), VictoryAxis (osy grafu) a *VictoryLine* (vykreslení dat). Obrazovka, která tuto knihovnu používá je popsána v kapitole 4.1.6.6.

4.1.6 Komponenty aplikace

V této kapitole jsou popsány všechny komponenty aplikace. U každé z nich je obrázek toho, jak vypadá na Android a iOS verzi aplikace. Všechny komponenty se nachází ve složce *Components* popsané v kapitole 4.1.4.

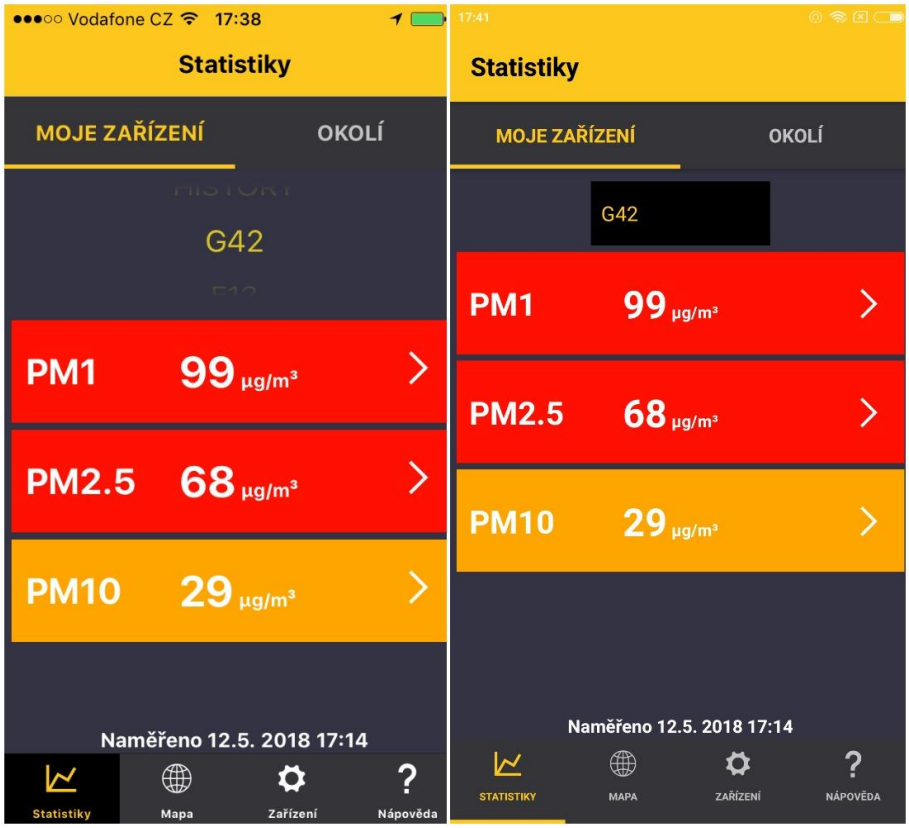
4.1.6.1 Statistiky

Komponenta statistik se nachází v souboru *statistics.js*. Slouží k vykreslení naměřených hodnot částic jednotlivých zařízení uživatele. Každá částice s její odpovídající hodnotou tvoří řádek ve FlatListu[17], což je React Native komponenta pro vytváření seznamů s více položkami. Jednotlivé řádky jsou zabarveny dle toho, jestli hodnota částice překračuje limity (červená – překročení denního limitu, oranžová – překročení ročního limitu, zelená – v pořádku). Na každý řádek lze také kliknout, tím se uživatel dostane na obrazovku s grafem, která je popsána v kapitole 4.1.6.6. Seznam hodnot jde táhnutím směrem dolů aktualizovat. Dále obrazovka obsahuje Picker[18], ve kterém uživatel volí zařízení, jehož hodnoty se mají ve FlatListu zobrazit a ve spodní části obrazovky je pak informace, kdy byly tyto hodnoty naměřeny. Komponenta naslouchá změnám globálního *store*, takže když uživatel přidá zařízení, jeho hodnoty se automaticky stáhnou a zobrazí. Obrazovka komponenty se nachází na Obrázek 12: Obrazovka statistik, iOS (vlevo) a Android (vpravo).

[16] https://github.com/FormidableLabs/victory-native

[17] https://facebook.github.io/react-native/docs/flatlist.html

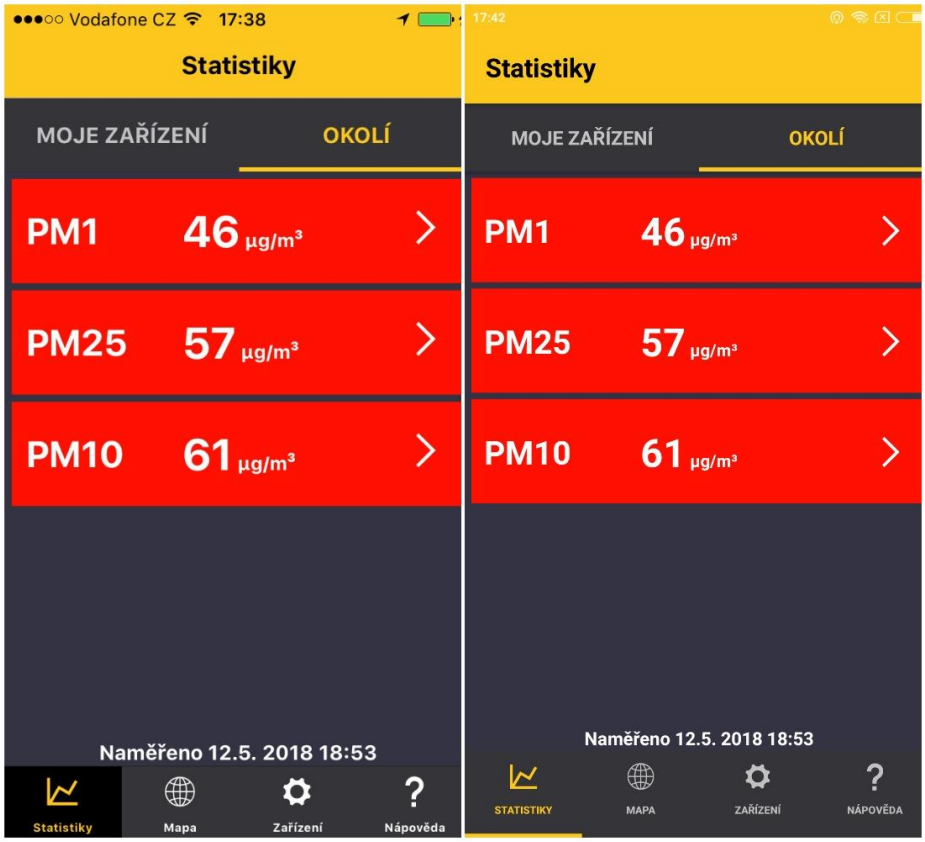
[18] https://facebook.github.io/react-native/docs/picker.html



Obrázek 12: Obrazovka statistik, iOS (vlevo) a Android (vpravo)

4.1.6.2 Lokální statistiky

Komponenta lokálních statistik se nachází v souboru *statisticsLocal.js.* Její podoba a chování je totožná s komponentou statistik z předchozí kapitoly 4.1.6.1 s tím rozdílem, že neobsahuje Picker. Smysl této obrazovky je uživateli ukazovat statistiky například z okresu či kraje. Momentálně se na obrazovce zobrazují hodnoty jen z jediného lokálního zařízení, nicméně do budoucna by se zde mohl také přidat Picker a uživatel by pak mohl přepínat mezi různými krajskými či okresními stanicemi. Obrazovka komponenty se nachází na Obrázek 13: Obrazovka lokálních statistik, iOS (vlevo) a Android (vpravo).

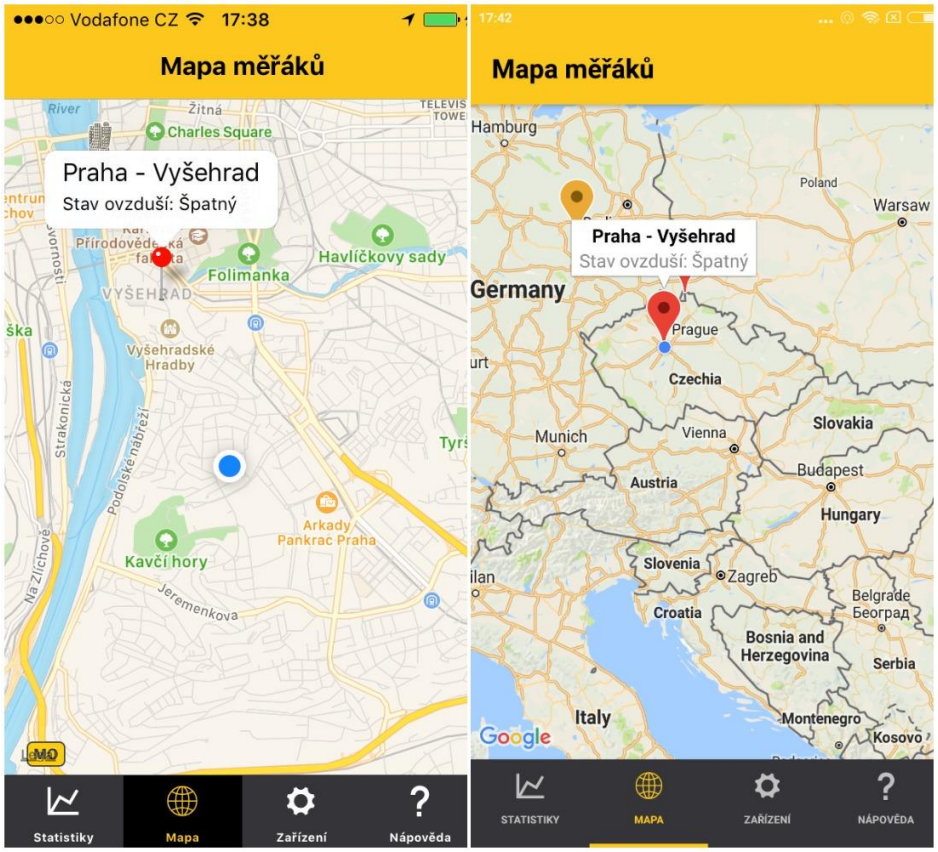


Obrázek 13: Obrazovka lokálních statistik, iOS (vlevo) a Android (vpravo)

4.1.6.3 Mapa měřáků

Komponenta mapy měřáků se nachází v souboru *devicesMap.js* a byla implementována nad rámec zadání. Pro vykreslení mapy využívá knihovnu react-native-maps[19]. Jejím smyslem je zobrazit uživateli všechny měřáky, které jsou pro něj na serveru dostupné (tedy nejen uživatelovy), společně s jejich stavem. Mapa obsahuje takzvané markery, které reprezentují jednotlivé měřáky. Jejich zabarvení záleží na stavu ovzduší v jeho okolí (červená – špatný, oranžová – uspokojivý, zelená – dobrý). Po zvolení daného měřáku se nad ním objeví bublina, kde je slovní popis stavu kvality ovzduší. Po doteku na bublinu se objeví modální okno s detailními naměřenými hodnotami zařízení. Při počátečním načtení se komponenta uživatele zeptá na povolení zjistit polohu telefonu a pokud to uživatel povolí, je mapa přiblížena na jeho polohu (toto ale zatím funguje pouze na iOS verzi). Jinak je počáteční region nastavený nad Prahu. Komponenta naslouchá na změny v globálním *store*, takže se průběžně aktualizují zařízení na mapě i s jejich hodnotami. Obrazovka komponenty se nachází na Obrázek 14: Obrazovka mapy zařízení, iOS (vlevo) a Android (vpravo).

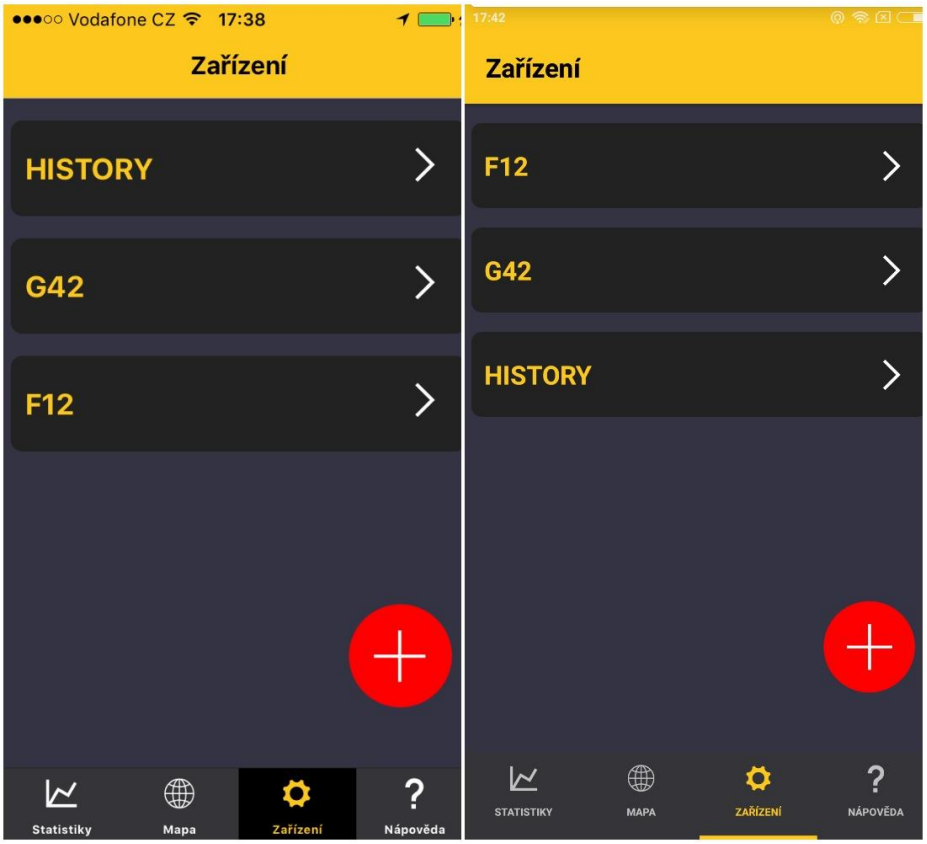
[19] https://github.com/react-community/react-native-maps



Obrázek 14: Obrazovka mapy zařízení, iOS (vlevo) a Android (vpravo)

4.1.6.4 Přehled zařízení

Komponenta přehledu zařízení se nachází v souboru *devices.js*. Obsahuje zařízení přidaná uživatelem ve formě FlatListu, kde každé lze rozkliknout. Tím se zobrazí obrazovka ovládání zařízení, která je popsána v kapitole 4.1.6.9. Pomocí červeného tlačítka ve spodní části obrazovky může uživatel přidat nové zařízení. Po kliknutí na toto tlačítko se objeví modální okno, kde je na výběr, jestli chce přidat zařízení ručně, zadáním kódu, nebo načíst QR kód. Přidání nového zařízení je popsáno v kapitolách 4.1.6.8 a 4.1.6.9. Podobně jako předchozí komponenty naslouchá tato komponenta na změny v globálním *store*, a když uživatel přidá nové zařízení na komponentě 4.1.6.8 či 4.1.6.9, aktualizuje seznam zařízení a překreslí obrazovku s nově přidaným zařízením. Obrazovka komponenty se nachází na Obrázek 15: Obrazovka zařízení uživatele, iOS (vlevo) a Android (vpravo)



Obrázek 15: Obrazovka zařízení uživatele, iOS (vlevo) a Android (vpravo)

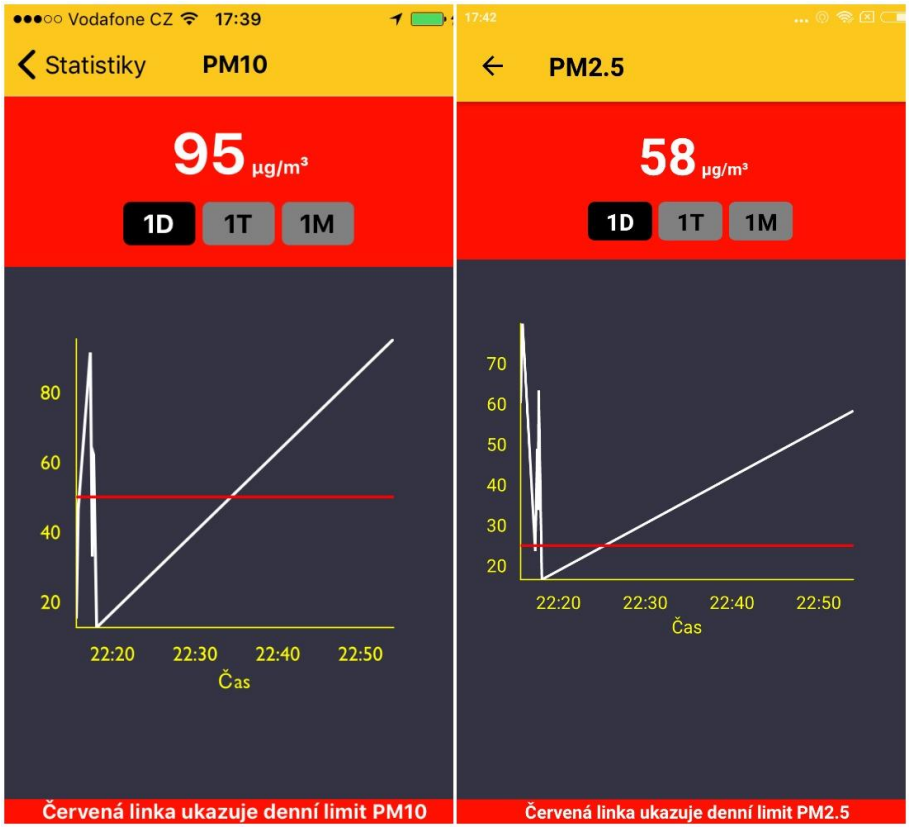
4.1.6.5 Nápověda

Komponenta nápovědy se nachází v souboru *help.js*. Jejím smyslem je informovat uživatele o jednotlivých částicích, které zařízení měří, a o jejich zdravotních rizicích a limitech. Obrazovka komponenty se nachází na Obrázek 16: Obrazovka nápovědy, iOS (vlevo) a Android (vpravo)

Obrázek 16: Obrazovka nápovědy, iOS (vlevo) a Android (vpravo)

4.1.6.6 Graf

Komponenta grafu se nachází v souboru *graph.js.* Graf zobrazuje historii naměřených hodnot daného zařízení a typu částice. V horní části obrazovky se zobrazuje aktuální hodnota dané částice a pozadí je zabarveno odpovídající barvou dle stavu (stejně jako odpovídající položka FlatListu na komponentě statistik v kapitole 4.1.6.1). Uživatel může přepínat mezi denním, týdenním a měsíčním grafem. Graf se vykreslí jen v případě, že existují alespoň 2 hodnoty dané částice v daném časovém intervalu. V grafu je červená linka znázorňující hodnotu denního limitu dané částice. Komponenta také naslouchá změnám v globálním *store,* a to z důvodu přepínání časového intervalu, kdy je potřeba stáhnout potřebná data ze serveru. Obrazovka komponenty se nachází na Obrázek 17: Obrazovka grafu, iOS (vlevo) a Android (vpravo)



Obrázek 17: Obrazovka grafu, iOS (vlevo) a Android (vpravo)

4.1.6.7 Přidání zařízení ručně

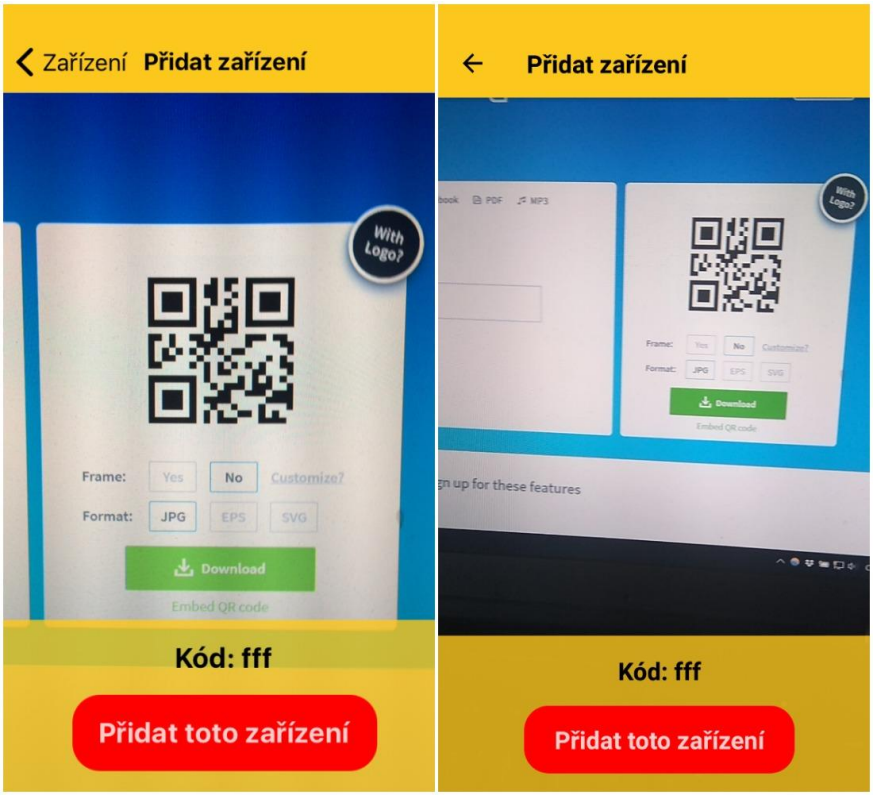
Komponenta přidání zařízení se nachází v souboru *deviceInput.js*. Obsahuje pole pro zadání jména zařízení a tlačítko pro jeho přidání. Po stisknutí tlačítka komponenta zavolá vrstvu controller pro komunikaci se serverem a ta ověří, zda dané zařízení existuje v databázi. Pokud ano, je zařízení přidáno do globálního *store* a následně také do paměti telefonu. Obrazovka komponenty se nachází na Obrázek 18: Obrazovka přidání zařízení, iOS (vlevo) a Android (vpravo)

Obrázek 18: Obrazovka přidání zařízení, iOS (vlevo) a Android (vpravo)

4.1.6.8 Přidání zařízení načtením QR kódu

Komponenta přidání zařízení pomocí QR kódu se nachází v souboru QRScanner.js a její fungování při přidávání zařízení je stejné jako fungování komponenty z předchozí kapitoly 4.1.6.7. Pro načítání QR kódu se používá knihovna BarCodeScanner[20], která od uživatele vyžaduje povolení používat kameru telefonu. Obrazovka komponenty se nachází na Obrázek 19: Obrazovka načtení zařízení QR kódem, iOS (vlevo) a Android (vpravo)

[20] https://docs.expo.io/versions/v27.0.0/sdk/bar-code-scanner



Obrázek 19: Obrazovka načtení zařízení QR kódem, iOS (vlevo) a Android (vpravo)

4.1.6.9 Ovládání zařízení

Komponenta ovládání zařízení se nachází v souboru *deviceControl.js.* Obsahuje tlačítka pro vypnutí a zapnutí měřáku, které zatím (měřák totiž neexistuje, nebylo tedy jak implementovat tyto funkce) jen mění text nadpisu ze „Zapnuto“ na „Vypnuto“ a naopak. Tlačítko s nápisem „Vymazat toto zařízení“ ovšem funguje a smaže dané zařízení z globálního *store* a paměti telefonu. Obrazovka komponenty se nachází na Obrázek 20: Obrazovka ovládání zařízení, iOS (vlevo) a Android (vpravo)

Obrázek 20: Obrazovka ovládání zařízení, iOS (vlevo) a Android (vpravo)

4.1.7 Ukládání a čtení z paměti telefonu

Ukládání a načítání stavu z paměti je řešeno pomocí třídy *AsyncStorage.* Jedná se o systém ukládání do paměti ve formě klíč – hodnota, který je globální pro celou aplikaci, je asynchronní a není šifrovaný. Na platformě iOS je ukládání řešeno pomocí serializovaného slovníku a separátních souborů v případě paměťově náročných hodnot. Na platformě Android používá *AsyncStorage* RocksDB nebo SQLite podle toho, co je dostupné [8].

V naší aplikaci se do *AsyncStorage* ukládají uživatelova zařízení, které se nacházejí v globálním *store* (ten je popsán v kapitole 4.1.5.2). K uložení do paměti dojde vždy, když uživatel přidá zařízení pomocí QR kódu nebo ručního zadání tohoto kódu, a také v případě, že uživatel smaže zařízení. Zařízení se ukládají pouze s jejich jménem (kódem) bez jakýchkoliv dalších dat.

Při spuštění aplikace se z *AsyncStorage* načítají zařízení uživatele. Po jejich načtení se aplikace pokusí získat aktuální data všech přidaných zařízení a také lokální stav ze serveru.

4.2 Server

V této kapitole je popsána implementace serverové části této práce. Jsou zde také popsány použité technologie.

4.2.1 Node.js

Pro vytvoření serveru byl zvolen open-source framework Node.js verze 6.11.4, který je nástavbou V8 engine[21] od společnosti Google, na kterém také například běží prohlížeč Google Chrome. Server-side aplikace Node.js jsou psány v jazyce JavaScript a mezi jejich hlavní výhody patří to, že veškerá volání API jsou asynchronní (server nečeká na odpověď, ta se předává pomocí callback funkce, což je funkce, která se zavolá v momentě obdržení odpovědi od API), jednotlivá volání se tedy vzájemně neblokují. Jelikož je V8 engine napsaný v jazyce C++, je vykonávání kódu také velmi rychlé.

4.2.2 REST API

Vedoucím práce byl stanoven jediný požadavek na server – aby dodržoval architekturu typu REST. REST rozhraní je takové rozhraní, které umožňuje přístup a manipulaci s daty pomocí operací CRUD (Create, Read, Update, Delete). Tyto operace musí být bez stavové (ani klient ani server si o sobě nepamatují informace). Náš server potřebuje a má proto implementovány pouze operace Create, kterou využívá generátor dat k uložení dat do databáze, a Read, kterou využívá aplikace pro čtení těchto dat.

4.2.3 Implementace serveru

Server byl vytvořen ve dvou verzích. První verze slouží k vytvoření serveru na lokální síti a druhá verze byla pak vytvořena pro spouštění na hostitelské platformě. Pro druhou verzi byla využita platfoma Heroku[22], která je zdarma a podporuje Node.js. Lokální server poslouchá na portu 8080. Oba servery používají framework Express[23] pro vytvoření směrovací tabulky, která pak provádí akce dle typu HTTP dotazu a URL odkazu. Express se formou callback funkce stará o zaslání odpovědi serveru klientovi poté, co je obdržena odpověď z databáze.
Pro HTTP dotaz typu POST naslouchá server na následující URL:

- http://ipadresa:port/setStats?parametr1=hodnota&parametr2=hodnota&... – Kde parametry URL odkazu jsou jednotlivé naměřené hodnoty měřáku a jeho název. Dotaz slouží k zavolání metody *setStats,* která parametry odkazu uloží do databáze k příslušnému měřáku.

Pro HTTP dotazy typu GET naslouchá server na následující URL:

- http://ipadresa:port/getStats/zarizeni – Kde *zarizeni* v dotazu je jméno měřáku. Dotaz slouží k zavolání metody *getStats,* která získá všechny naměřené hodnoty daného měřáku z databáze a zašle je v odpovědi klientovi.
- http://ipadresa:port/getCurrentStats/zarizeni – Kde *zarizeni* v dotazu je jméno měřáku. Dotaz slouží k zavolání metody *getCurrentStats,* která získá nejaktuálnější naměřené hodnoty daného měřáku z databáze a zašle je v odpovědi klientovi.
- http://ipadresa:port/getDevice/zarizeni – Kde *zarizeni* v dotazu je jméno měřáku. Dotaz slouží k zavolání metody *getDevice,* která zjistí z databáze, zda dané zařízení existuje. Pokud ano, vrátí v odpovědi *true.* Pokud ne, vrátí *false*.
- http://ipadresa:port/getPositions– Dotaz slouží k zavolání metody *getPositions,* která získá z databáze všechny měřáky včetně jejich GPS souřadnic, názvů, lokací a posledních naměřených hodnot.

Odpovědi serveru s daty se zasílají ve formátu JSON, který je v dnešní době velmi rozšířený a používá k zápisu objektovou notaci JavaScriptu, v němž je napsán i náš server a mobilní aplikace.

[21] https://developers.google.com/v8/
[22] https://heroku.com/
[23] https://expressjs.com/

4.3 Generátor dat

Generátor dat byl napsán ve skriptovacím jazyce Python verze 3.4. Slouží k simulaci chování měřáku spojeného s aplikací. Používá knihovnu requests[24] pro vytvoření HTTP dotazu typu POST a zaslání dat na server jako parametry tohoto dotazu. Data se generují každých 15 sekund a skládají se z naměřených hodnot PM1, PM10, PM2.5, dále data a času vygenerování a názvu zařízení, které je naměřilo.

4.4 Databáze

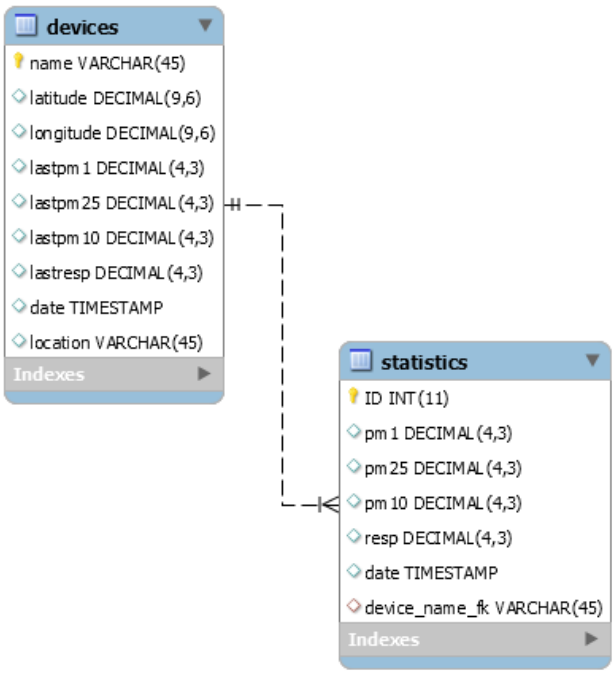
Databáze byla stejně jako server vytvořena dvakrát, jednou pro lokální verzi serveru a podruhé na platformě Heroku. Lokální databáze používá MySQL a databáze na Heroku je typu Postgresql. Databáze slouží pro ukládání dat z měřáku (generátoru dat), které jsou zasílány ze serveru. Na Obrázek 21: Datový model databáze je znázorněn její datový model v MySQL (Postgresql model se liší pouze v názvu datových typů), ve kterém jsou dvě tabulky, a to tabulka měřáků (devices) včetně jejich GPS souřadnic (latitude, longitude), názvů a tabulka naměřených dat (statistics) daného měřáku.

Do tabulky devices byly nakonec z optimalizačních důvodů přidány sloupečky posledních naměřených hodnot (lastpm1, lastpm2 atd.). Server pak při přijetí nových dat z generátoru tyto data uloží jak do tabulky statistik, tak do tabulky měřáku. Nejnovější data daného zařízení se tedy vyskytují v databázi dvakrát, jednou v tabulce statistik a jednou v tabulce měřáků. Důvodem je to, že při zobrazení mapy se všemi dostupnými měřáky v aplikaci se musí stahovat nejnovější data každého z nich. Tedy dotaz na databázi by musel najít zařízení a pak prohledat tabulku statistik a najít jeho nejnovější údaje. Jelikož řádků v tabulce statistik bude velké množství a v každém případě mnohem víc, něž v tabulce měřáků, je rychlejší, když tyto nejnovější údaje jsou v databázi vždy uloženy u každého měřáku.

Sloupec location (lokace) v tabulce devices je název místa na daných souřadnicích měřáku. Tento název se dá například najít pomocí Reverse Geocoding[25], což je API od společnosti Google, kterému vývojář zadá GPS souřadnice a API vrátí název lokace na těchto souřadnicích. Sloupec je zatím zadáván ručně, jelikož jej autor práce nestihl implementovat a nebylo to ani prioritou. Tato lokace slouží jako název měřáku v mapě měřáků aplikace.

[24] http://docs.python-requests.org/en/master/

[25] https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse



Obrázek 21: Datový model databáze

5. Uživatelské testování aplikace

Cílem testování bylo zjistit, zda je aplikace dostatečně jednoduchá a intuitivní v ovládání tak, že uživatel nemá problém dosáhnout svého záměru. V testování bylo hodnoceno zejména to, jak dlouho uživateli zabralo splnit zadaný úkol, dále srozumitelnost navigace v rámci aplikace, tedy jestli našel přímou cestu k hledané funkci, nebo zaváhal a hledal ji například na jiné obrazovce.

5.1 Průběh testování

V této kapitole je popsáno, jak testování aplikace s uživateli probíhalo.

5.1.1 Způsob testování

Testování se konalo v místnosti osvětlené denním světlem se stolem, na kterém měl uživatel připraven telefon platformy iOS (iPhone 5s s verzí operačního systému iOS 10.3.3) nebo Android (Xiaomi Redmi Note 4 Global s verzí operačního systému Android 7.0) spuštěný na úvodní obrazovce aplikace (statistiky) a tablet, na kterém byl zobrazen QR kód. Nejdříve uživatel dostal pre-test dotazník, následně papír s úvodem o aplikaci, po jehož přečtení dostal uživatel papír s testovacími scénáři, které měl za úkol plnit v pořadí, jak šly za sebou. Na konci testování dostává účastník obyčejně post-test dotazník, nicméně při testování naší aplikace byl s účastníky proveden rozhovor o jejich dojmech z aplikace.

5.1.2 Výběr účastníků testování

Pro testování byli vybráni lidé vlastnící chytrý telefon s operačním systémem Android či iOS, a protože aplikace cílí na běžného uživatele, byli účastníci vybíráni pokud možno ze všech věkových kategorií a schopností ovládání telefonu. Aplikaci celkem otestovalo 5 lidí, přičemž 3 z nich byli primární uživatelé Android a 2 z nich byli primárními uživateli iOS. Při testování dostal každý účastník telefon se stejným operačním systémem jako byl jeho primární telefon.

5.1.3 Pre-test dotazník

Pre-test dotazník dostal účastník na začátku testu. Sloužil k zaznamenání informací o testované osobě, které byly pro testování relevantní. Seznam otázek dotazníku je v seznamu níže.

1. Jaký operační systém má Váš primární mobilní telefon?
    a. iOS
    b. Android
2. Jaká je úroveň vašich schopností v ovládání telefonu?
    a. běžný uživatel
    b. pokročilý uživatel
    c. zkušený uživatel
3. Kolik je Vám let?
4. Jaké je vaše pohlaví.
    a. žena
    b. muž
5. Souhlasíte se zaznamenáním popisu průběhu testu a použití vašich odpovědí v rámci vypracování bakalářské práce na ČVUT FEL?
    a. ano
    b. ne

5.1.4 Úvod o aplikaci

Po vyplnění pre-test dotazníku měl uživatel za úkol přečíst si následující text, který jej měl uvést do kontextu aplikace.

„Protože Vás zajímala kvalita a stav ovzduší ve Vašem okolí, koupili jste si zařízení, které tyto údaje měří. K zobrazování aktuálních hodnot naměřených zařízením jste používali webovou aplikaci, nicméně po pár dnech jste zjistili, že k zařízení existuje i mobilní aplikace, na které můžete také vidět aktuální hodnoty stavu ovzduší. Rozhodli jste se ji proto vyzkoušet“.

5.1.5 Testované scénáře

Scénář měl za úkol pokrýt nejčastější použití aplikace. Protože pro přidání zařízení musí existovat zařízení se stejným jménem v databázi, měl uživatel ve scénářích pevně zadány zařízení s názvy „Zarizeni1“ a „Zarizeni2“, kde druhé jmenované bylo formou QR kódu. Obě zařízení měly v databázi dopředu vygenerovány naměřené hodnoty. Jednotlivé scénáře ve stejném pořadí, jako je měl plnit uživatel, se nachází v seznamu níže.

1. Zjistěte, jaký je stav (statistika) ovzduší ve Vašem okolí.
2. Přidejte zařízení do aplikace pomocí ručního zadání kódu „Zarizeni1“ (bez uvozovek).
3. Přidejte zařízení „Zarizeni2“ do aplikace pomocí načtení QR kódu.
4. Zjistěte aktuální naměřené hodnoty (statistiky) zařízení „Zarizeni1“.
5. Zjistěte, co znamená údaj „PM2.5“ a popište slovy, jaký má částice PM2.5 vliv na Vaše zdraví.
6. Zjistěte aktuální hodnotu „PM2.5“ naměřenou zařízením „Zarizeni2“.
7. Aktualizujte stav (statistiky) zařízení „Zarizeni2“.
8. Popište slovy, který z limitů (denní / roční) naměřená hodnota částice „PM2.5“ zařízením „Zarizeni2“ překračuje.
9. Zobrazte týdenní graf částice „PM10“ zařízení „Zarizeni1“.
10. Zobrazte mapu všech zařízení a zobrazte naměřené hodnoty zařízení nacházejícího se na mapě v Praze.
11. Vypněte zařízení „Zarizeni2“.
12. Odstraňte zařízení „Zarizeni2“ z aplikace.

5.1.6 Post-test rozhovor

Tento rozhovor proběhl na konci testování. Každému z uživatelů byly postupně kladeny otázky, které se nachází v seznamu níže.

1. Co se vám na aplikaci líbilo? Zaujalo vás něco?
2. Co se Vám na aplikaci nelíbilo? Změnili byste něco?

5.2 Vyhodnocení testování

V této kapitole jsou zaznamenány jednotlivé odpovědi účastníků, dále shrnutí jejich průchodu aplikací a závěrečného rozhovoru. Na konci kapitoly je pak přehled nálezů a změny v aplikaci snažící se reagovat a řešit tyto nálezy.

5.2.1 Účastník 1

Účastník 1 byl testován na telefonu s operačním systémem Android.

- Pre-test dotazník
    1. Android
    2. Běžný uživatel
    3. 50 let
    4. muž
    5. Ano

- Průchod scénáři
    1. Bez problému.
    2. Bez problému.
    3. Bez problému.
    4. Při tomto úkolu účastník zaváhal. Po přidání zařízení do aplikace byl automaticky přesměrován zpět na obrazovku s přehledem zařízení a statistiku zařízení „Zarizeni1“ hledal rozkliknutím daného zařízení v tomto přehledu, to nicméně vede na obrazovku s ovládáním daného zařízení. Po chvíli ale účastník našel správnou obrazovku.
    5. Při tomto úkolu došlo také k zaváhání. Popis částice PM2.5 se snažil najít kliknutím na údaj PM2.5 na obrazovce statistik. Nicméně to vede na obrazovku s grafem a historií naměřených údajů. Po chvíli ale účastník správně našel záložku „Nápověda“ v navigaci a úkol splnil.
    6. Účastník nevěděl, jak změnit zařízení na obrazovce statistik. Tento úkol prováděl asi minutu, nakonec se mu jej ale zdárně podařilo splnit.
    7. Bez problému.
    8. Bez problému.
    9. Bez problému.
    10. Bez problému.
    11. Bez problému.
    12. Bez problému.
- Post-test rozhovor
    1. Účastníkovi se líbilo celkové provedení aplikace.
    2. Účastníkovi se nelíbilo, že z obrazovky mapy zařízení se nelze přesunout pomocí swipe (pohyb prstu doleva či doprava po obrazovce) na vedlejší obrazovky navigace Tab Bar. Také aplikaci vytkl přepínaní mezi jednotlivými zařízeními aplikace na obrazovce statistik.

5.2.2 Účastník 2

Účastník 2 byl testován na telefonu s operačním systémem Android.

- Pre-test dotazník
    1. Android
    2. Běžný uživatel
    3. 51 let
    4. žena
    5. Ano
- Průchod scénáři
    1. Bez problému.
    2. Bez problému.
    3. Bez problému.
    4. Účastník 2 při tomto úkolu stejně jako předchozí účastník zaváhal a po přidání zařízení do aplikace hledal jeho statistiku rozkliknutím daného zařízení v přehledu zařízení. Po chvíli ale účastník také zdárně našel správnou obrazovku.
    5. Účastník se opět jako předchozí uživatel snažil najít údaje o PM2.5 kliknutím na údaj PM2.5 na obrazovce statistik. Po chvíli správně našel záložku „Nápověda“ v navigaci a úkol splnil.
    6. Účastník nevěděl, jak změnit zařízení na obrazovce statistik. Tento úkol se uživateli nepodařilo splnit a do testování bylo potřeba zasáhnout a pomoci s dokončením úkolu.

7. Účastník nevěděl, jak aktualizovat hodnoty zařízení (potáhnutím seznamu hodnot prstem směrem dolů). Do testování bylo opět potřeba zasáhnout a pomoci s úkolem.
8. Bez problému.
9. Bez problému.
10. Bez problému.
11. Bez problému.
12. Bez problému.

- Post-test rozhovor
    1. Účastníkovi se líbil grafický vzhled aplikace, nicméně doporučil přidat do nápovědy informace o smyslu jednotlivých barevných odlišení naměřených hodnot na obrazovce statistik a jinde v aplikaci.
    2. Nelíbilo se mu provedení změny zařízení a způsob aktualizace hodnot na obrazovce statistik.

5.2.2 Účastník 3

Účastník 3 byl testován na telefonu s operačním systémem Android.

- Pre-test dotazník
    1. Android
    2. Pokročilý uživatel
    3. 24 let
    4. žena
    5. Ano
- Průchod scénáři
    1. Bez problému.
    2. Bez problému.
    3. Bez problému.
    4. Účastník 3 při tomto úkolu stejně jako předchozí účastníci zaváhal a po přidání zařízení do aplikace hledal jeho statistiku rozkliknutím daného zařízení v přehledu zařízení. Také ale po chvíli úkol splnil.
    5. Účastník 3 se jako předchozí uživatelé snažil najít údaje o PM2.5 kliknutím na údaj PM2.5 na obrazovce statistik. Po chvíli ale úkol splnil.
    6. Účastník zde také zaváhal, ale po chvíli na změnu zařízení přišel a úkol splnil.
    7. Bez problému.
    8. Bez problému.
    9. Bez problému.
    10. Bez problému.
    11. Bez problému.
    12. Bez problému.
- Post-test rozhovor
    1. Účastníkovi se líbil grafický vzhled aplikace.
    2. Doporučil zlepšit provedení změny zařízení na obrazovce statistik.

5.2.3 Účastník 4

Účastník 4 byl testován na telefonu s operačním systémem iOS.

- Pre-test dotazník
    1. iOS
    2. Pokročilý uživatel
    3. 18 let
    4. muž
    5. Ano

- Průchod scénáři
    1. Bez problému.
    2. Bez problému.
    3. Bez problému.
    4. Účastník 4 hledal tyto statistiky na stejné obrazovce jako všichni jeho předchůdci. Rychle ale reagoval a správnou obrazovku našel.
    5. Účastník 4 hledal údaje o částici PM2.5 na stejné obrazovce jako všichni jeho předchůdci. Opět ale rychle reagoval a správnou obrazovku našel.
    6. Bez problému.
    7. Bez problému.
    8. Bez problému.
    9. Bez problému.
    10. Bez problému.
    11. Bez problému.
    12. Bez problému.
- Post-test rozhovor
    1. Účastník kladně hodnotil grafické provedení aplikace.
    2. Neměl žádné výtky.

5.2.4 Účastník 5

Účastník 5 byl testován na telefonu s operačním systémem iOS.

- Pre-test dotazník
    1. iOS
    2. Běžný uživatel
    3. 58 let
    4. muž
    5. Ano
- Průchod scénáři
    1. Bez problému.
    2. Bez problému.
    3. Bez problému.
    4. Účastník 5 hledal tyto statistiky na stejné obrazovce jako všichni jeho předchůdci. Správnou obrazovku po chvíli našel.
    5. Účastník 5 hledal údaje o částici PM2.5 na stejné obrazovce jako všichni jeho předchůdci. Také ale po chvíli správnou obrazovku našel.
    6. Při změně zařízení došlo k minimálnímu zaváhání.
    7. Účastníkovi nebylo jasné, jak hodnoty aktualizovat. Po chvíli ale úkol splnil.
    8. Bez problému.
    9. Bez problému.
    10. Bez problému.
    11. Bez problému.
    12. Bez problému.
- Post-test rozhovor
    1. Účastníkovi se líbilo grafické provedení aplikace, ale doporučil změnit barvu pozadí grafu na světlejší odstín.
    2. Účastník navrhnul přidání ikonky aktualizace hodnot na obrazovku statistik.

5.2.6 Nálezy

Nálezy byly rozděleny dle priority na 3 úrovně. Rozdělení se nachází v seznamu níže.

1. Nízká priorita – jedná se o nálezy, které uživateli sice vadí, nicméně mu nebrání v používání aplikace a nejsou pro něj frustrující.

2. Střední priorita – jedná se o nálezy, které negativně ovlivňují používání aplikace a vzbuzují v uživateli frustraci. Příliš mnoho problémů z této kategorie může uživatele odradit od používání aplikace.
3. Vysoká priorita – problémy z této kategorie zásadně a negativně ovlivňují používání aplikace. Tyto problémy by měly být odstraněny co nejdříve.

Nedostatky s vysokou prioritou byly do této kategorie zařazeny v případě, že alespoň jednomu účastníkovi zabránily ve vykonání úkolu.

V Tabulka 2: Nálezy při testování aplikace jsou vypsány nálezy, které vyvstaly při testování. U každého z nich se nachází platforma, na které byl problém nalezen, dále popis problému a doporučené řešení. Nálezy jsou seřazeny dle priority od nejnižší po nejvyšší.

Tabulka 2: Nálezy při testování aplikace

Priorita	Platforma	Popis nálezu	Doporučené řešení
nízká	iOS i Android	Účastníkovi 5 testu se nelíbila tmavá barva pozadí grafu na obrazovce grafu.	Změnit barvu pozadí grafu na bílou.
nízká	iOS i Android	V aplikaci není na první pohled jasná logika barevných pozadí a barev textu naměřených hodnot v rámci aplikace.	Přidat do nápovědy význam jednotlivých barev.
nízká	iOS i Android	Umístění záložky „Mapa“ v navigaci aplikace uprostřed není vhodné. Narušuje to plynulost navigace Tab Bar, protože se z mapy nejde dostat na vedlejší obrazovky pomocí swipe (táhnutí prstu doleva či doprava po obrazovce).	Umístit záložku „Mapa“ jako poslední v panelu Tab Bar.
vysoká	iOS i Android	Některým účastníkům nebylo jasné, jak aktualizovat hodnoty zařízení na obrazovce statistik. Účastníkovi 2 tím bylo znemožněno provedení úkolu.	Přidat ikonku aktualizace na obrazovku statistik.
vysoká	Android	Změna zařízení na obrazovce statistik nebyla účastníkům testování jasná. Účastníkovi 2 tím bylo znemožněno provedení úkolu.	Změnit grafické provedení změny zařízení.

5.2.7 Změny v aplikaci

Všechny nalezené problémy v předchozí kapitole 5.2.6 byly po provedení testování na základě doporučených řešení opraveny.

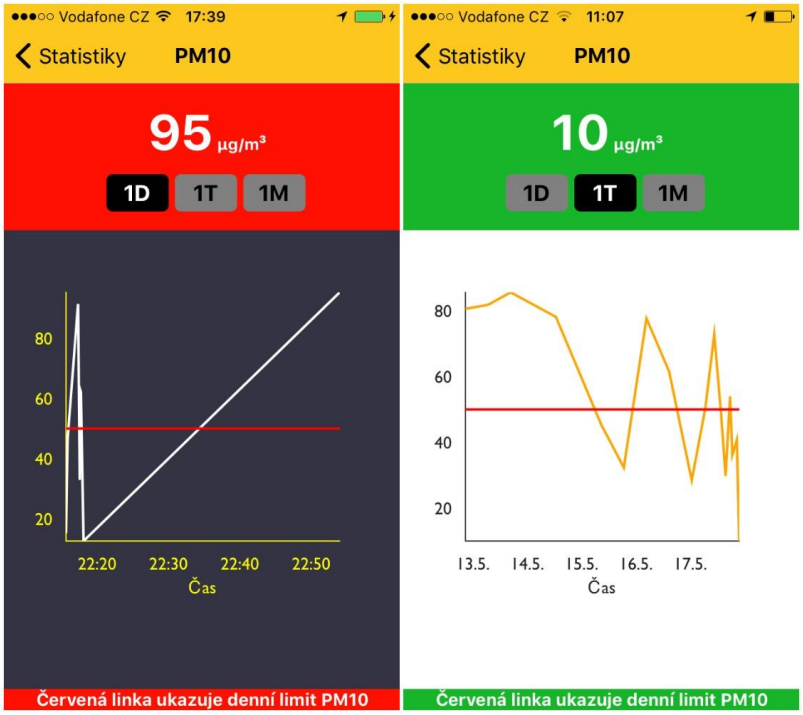
První opravou bylo přidání popisu významu barev v nápovědě na obou platformách. Obrázek 22: Původní obrazovka nápovědy (vlevo) a oprava (vpravo) obsahuje srovnání předchozího řešení a jeho opravy.

Obrázek 22: Původní obrazovka nápovědy (vlevo) a oprava (vpravo)

Následně byla přemístěna záložka „Mapa“ v Tab Bar navigaci na poslední pozici na obou platformách. Obrázek 23: Původní lišta Tab Bar navigace (vlevo) a oprava (vpravo) obsahuje srovnání předchozího řešení a opravy.

Obrázek 23: Původní lišta Tab Bar navigace (vlevo) a oprava (vpravo)

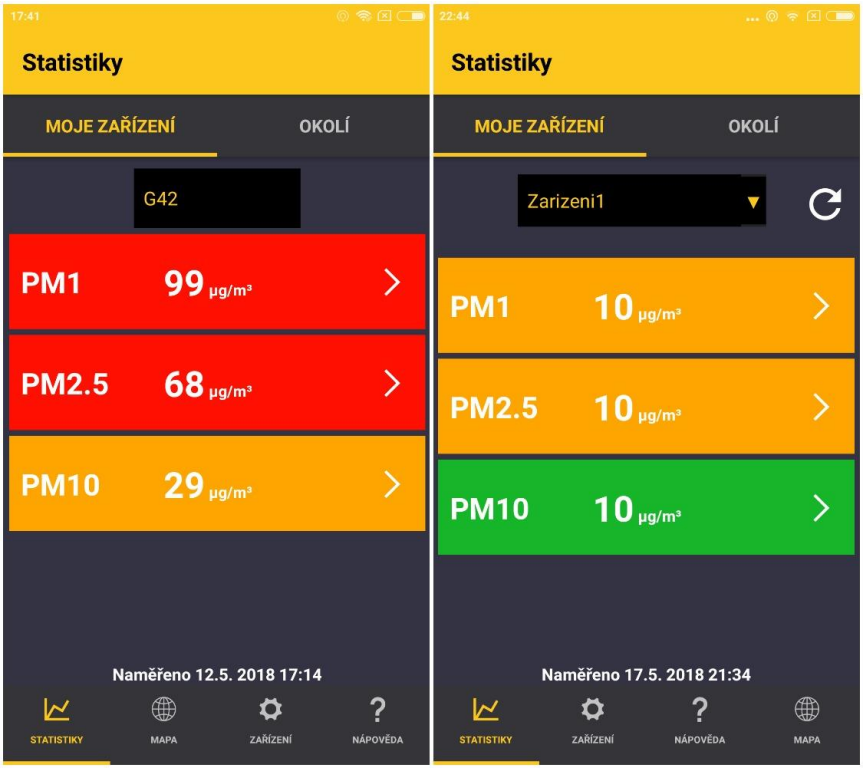
Na obou platformách bylo změněno pozadí grafu na bílou místo šedé. Barevně to k aplikaci více pasuje. Obrázek 24: Původní obrazovka grafu (vlevo) a její oprava (vpravo) obsahuje srovnání předchozího řešení a opravy.



Obrázek 24: Původní obrazovka grafu (vlevo) a její oprava (vpravo)

Na obou platformách byla do obrazovky statistik přidána ikonka aktualizace nad seznam naměřených hodnot napravo od výběru zařízení. Obrázek 25: Původní obrazovka statistik (vlevo) a její oprava (vpravo) ukazuje srovnání předchozího řešení a opravy.

Nakonec byl na platformě Android přidán do Pickeru pro výběr zařízení takzvaný spinner, což je v tomto případě rovnostranný trojúhelník, který směřuje špičkou dolů. Na platformě Android je to standard pro políčka s výběrem z více možností. Obrázek 25: Původní obrazovka statistik (vlevo) a její oprava (vpravo) ukazuje srovnání předchozího řešení a opravy.



Obrázek 25: Původní obrazovka statistik (vlevo) a její oprava (vpravo)

5.2.8 Shrnutí testování s uživateli

Při testování se našly 2 závažné a 3 méně závažné nedostatky, které byly vzápětí opraveny. Ačkoliv ve dvou úkolech v průběhu testování hledali všichni účastníci bez výjimky danou funkci na špatné obrazovce, nepovažujeme to za nedostatek aplikace. Toto zmatení vyplynulo z pořadí úkolů a uživatel se takto splete jen jednou. Při každém dalším použití již ví, kde správnou funkci hledat.

6. Závěr

Cílem této práce bylo navrhnout a implementovat multiplatformní mobilní aplikaci pro vzdálenou správu měřícího zařízení kvality ovzduší na základě analýzy již existujících řešení, cílové skupiny a požadavků daných vedoucím práce.

V úvodu práce byla provedena analýza, ve které byla prozkoumána problematika kvality ovzduší a jejího vlivu na lidské zdraví. Dále byla v analytické části provedena rešerše již existujících aplikací, která sloužila pro inspiraci a ucelení toho, jak by mohla výsledná aplikace vypadat a poznatky z ní pak byly použity v návrhové části.

V návrhové části byla nejdříve kategorizována a popsána cílová skupina uživatelů aplikace a následně byly s ohledem na tuto cílovou skupinu stanoveny požadavky na aplikaci, server, databázi a generátor hodnot simulující chování měřáku ovzduší – daný měřák ovzduší totiž v průběhu psaní práce neexistoval. Poté byl proveden návrh architektury aplikace, serveru a systému jako celku. Nakonec byl navržen vzhled hlavních obrazovek aplikace.

Dle provedeného návrhu pak byla implementována samotná aplikace ve frameworku React Native, který byl zvolen vedoucím práce. Tento framework se ukázal jako vhodná volba, protože umožňuje psát jednu multiplatformní aplikaci, která funguje jako nativní aplikace jak na platformě Android, tak iOS. Další jeho výhodou je, že umožňuje vidět změny v aplikaci na zařízení okamžitě při změně a uložení souboru s kódem, což výrazně zrychluje a ulehčuje vývoj. Pro potřeby testování funkčnosti pak byl dle návrhu implementován také generátor hodnot, simulující chování měřícího zařízení, server, který slouží k ukládání dat a jejich zprostředkování aplikaci, a nakonec databáze, na kterou se ukládají data ze serveru.

V závěru pak byla aplikace otestována s uživateli na obou platformách. Při testování sice bylo zjištěno několik nedostatků aplikace s vysokou nebo nízkou prioritou, nicméně celkový dojem uživatelů z aplikace byl pozitivní. Na základě testování byly nalezené nedostatky obratem opraveny.

6.1 Budoucí vývoj

Jak už bylo řečeno, měřící zařízení a potažmo ani server zatím neexistují. Aplikace funguje s vlastním serverem a databází, a proto bude v budoucnu potřeba upravit její fungování dle rozhraní, které tento nový server bude poskytovat.

7. Zdroje

[1] Ambient air pollution: Pollutants. *World Health Organization* [online]. Geneva, Switzerland: World Health Organization [cit. 2018-05-19]. Dostupné z: http://www.who.int/airpollution/ambient/pollutants/en/

[2] *World health statistics 2016: Monitoring health for the SDGs* [online]. Geneva, Switzerland: World Health Organization, 2016, s. 80 [cit. 2018-05-19]. ISBN 9789240695696. Dostupné z: http://apps.who.int/iris/bitstream/10665/206498/1/9789241565264_eng.pdf

[3] Ambient (outdoor) air quality and health. In: *World Health Organization* [online]. Geneva, Switzerland: World Health Organization, 2018 [cit. 2018-05-19]. Dostupné z: http://www.who.int/en/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health

[4] Apple on Hamburger Menus. In: *Manbolo Blog* [online]. Manbolo, 2014 [cit. 2018-05-19]. Dostupné z: http://blog.manbolo.com/2014/06/30/apple-on-hamburger-menus

[5] REMEŠ, Jan. *Návrh aplikace pro měření znečištění ovzduší – projekt Kanárci* [online]. Technická 2, Praha 6 - Dejvice, Praha, 2014 [cit. 2018-05-19]. Dostupné z: https://dspace.cvut.cz/handle/10467/24951. Diplomová práce. České vysoké učení technické, Fakulta elektrotechnická, Otevřená informatika.

[6] EISENMAN, Bonnie. *Learning React Native: Building Native Mobile Apps with JavaScript* [online]. 2. vyd. Sebastopol: O'Reilly Media, 2017 [cit. 2018-05-20]. ISBN 9781491989142. Dostupné z: http://shop.oreilly.com/product/0636920085270.do

[7] Three Principles - Redux. *Redux* [online]. London, England: Dan Abramov, 2018 [cit. 2018-05-19]. Dostupné z: https://redux.js.org/introduction/three-principles

[8] AsyncStorage. React Native [online]. Menlo Park, California, United States of America: Facebook [cit. 2018-05-19]. Dostupné z: https://facebook.github.io/react-native/docs/asyncstorage.html

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

WHO – World Health Organization
CHOPN – Chronická Obstrukční Plicní Nemoc
ČHMÚ – Český Hydrometeorologický Úřad
REST – Representational State Transfer
JSON – JavaScript Object Notation
XML – Extensible Markup Language
HTTP – Hypertext Transfer Protocol
GPS – Global Positioning System
API – Application Programming Interface
URL – Uniform Resource Locator
SDK – Software Development Kit
CSS – Cascade Style Sheet
QR– Quick Response
UI – User Interface

Příloha A

A.1 Obsah přiloženého CD

- AirMeter – složka s projektem
- airmeter.apk – instalační soubor aplikace pro platformu Android
- readme.txt – popis instalace a použití
- thesis.pdf – text práce ve formátu pdf
- thesis.docx – text práce ve formátu docx

Příloha B

B.1 Instalační instrukce

Instalační instrukce jsou rozděleny do několika kategorií, dle záměru čtenáře práce.

a) Vyzkoušení aplikace

Pokud má čtenář zájem aplikaci pouze vyzkoušet, nachází se na přiloženém CD Android apk aplikace *airmeter.apk*, které stačí nainstalovat v mobilním telefonu (je třeba povolit instalaci neznámých aplikací). K použití aplikace stačí mít mobilní telefon připojený k internetu. K aplikaci byl vytvořen server a databáze na platformě Heroku, takže pro vyzkoušení fungování aplikace stačí přidat jedno ze zařízení dostupných v databázi – „Zarizeni1“, „Zarizeni2“ (bez uvozovek). Více informací o serveru a generování nových dat je na konci kapitoly.

b) Úpravy kódu a simulace aplikace na zařízení pomocí Expo

Pokud má čtenář zájem kód upravovat a simulovat na vlastním zařízení, nejjednodušší cestou je použití systému Expo, pomocí kterého byla aplikace vyvíjena. Je třeba nainstalovat Node.js, jehož součástí je také balíček npm. Dále je třeba spustit příkaz *npm install exp --global,* kde je třeba se také zaregistrovat na stránky Expo. Poté stačí přejít do složky *AirMeter* projektu a spustit příkaz *npm install*, který stáhne potřebné knihovny a závislosti jenž jsou popsané v souboru *package.json*. Dále stačí zadat příkaz *exp start,* který spustí instanci aplikace na počítači a vygeneruje QR kód. Je třeba nainstalovat aplikaci Expo v Google Play či App Store a QR kód naskenovat či ručně zadat, což následně aplikaci na zařízení otevře. Pokud se uživatel do mobilní aplikace Expo přihlásí, zobrazují se mu instance projektů běžících na počítači automaticky. Další možnosti Expo jsou popsány v dokumentaci[26].

c) Úpravy kódu a simulace aplikace bez pomoci Expo

Pokud má čtenář zájem kód upravovat a simulovat na vlastním zařízení bez použití Expo, musí nejdříve projekt odpojit od Expo příkazem *exp detach*. Instrukce jsou v dokumentaci[1] Expo. Tímto příkazem dojde k rozdělení projektu na Android a iOS část a projekt se předělá na strukturu, kterou by měl při vytvoření standardním příkazem *react-native init JmenoProjektu,* kterým se vytvářejí React Native aplikace. Další postup je popsán na stránkách React Native[27].

Server i s databází byl pro čtenáře vytvořen na stránce *https://mysterious-cliffs-74298.herokuapp.com*, aplikace i generátor dat mají tuto adresu pevně nastavenou (aplikace v souboru *constants.js*, generátor pak v hlavičce kódu) a čtenář práce tedy nemusí nic nastavovat, stačí si nainstalovat aplikaci. Databáze na této stránce obsahuje 4 zařízení se jmény LOCAL (zařízení s naměřenými daty v okolí), Zarizeni1, Zarizeni2 a F12. Tyto zařízení lze do aplikace přidat. Některá zařízení mají vygenerovaných více hodnot (např. Zarizeni1), nicméně pouze v měsíci květnu roku 2018. Pokud má uživatel zájem vidět graf hodnot za poslední den / týden / měsíc, musí hodnoty pro zařízení vygenerovat pomocí generátoru dat.

Server je možno vytvořit lokálně i s vlastní databází MySQL (případně jinou, ale je třeba změnit také v kódu serveru), pak je ale třeba změnit adresu serveru v aplikaci a generátoru dat. Zdrojový kód serveru se nachází v souboru *server.js* a spouští se příkazem *npm server*.

Generátor dat se nachází v souboru *generator.py*. Pro spuštění je třeba mít nainstalovaný Python. Pro použití stačí v hlavičce kódu nastavit jméno zařízení (proměnná *device*), spodní / horní hranici generovaných hodnot (proměnné *lowerBound* a *upperBound)* a nakonec počet hodin, které se mají odečíst od současného data (proměnná *hoursOffset* – *pokud* je její hodnota 0, vygeneruje se současné datum).

[26] https://docs.expo.io/versions/v27.0.0/introduction/

[27] https://facebook.github.io/react-native/docs/getting-started.html